electronics

MDPI

Article

Compact Implementation of ARIA on 16-Bit MSP430 and 32-Bit ARM Cortex-M3 Microcontrollers

Hwajeong Seo *, Hyunjun Kim, Kyoungbae Jang, Hyeokdong Kwon, Minjoo Sim, Gyeongju Song and Siwoo Uhm

Division of IT Convergence Engineering, Hansung University, Seoul 02876, Korea; amdjd0704@hansung.ac.kr (H.K.); starj1234@hansung.ac.kr (K.J.); hyeok@hansung.ac.kr (H.K.); alswntla@hansung.ac.kr (M.S.); thdrudwn98@hansung.ac.kr (G.S.); smile267@hansung.ac.kr (S.U.)
* Correspondence: hwajeong@hansung.ac.kr; Tel.: +82-760-8033

Abstract: In this paper, we propose the first ARIA block cipher on both MSP430 and Advanced RISC Machines (ARM) microcontrollers. To achieve the optimized ARIA implementation on target embedded processors, core operations of ARIA, such as substitute and diffusion layers, are carefully re-designed for both MSP430 (Texas Instruments, Dallas, TX, USA) and ARM Cortex-M3 microcontrollers (STMicroelectronics, Geneva, Switzerland). In particular, two bytes of input data in ARIA block cipher are concatenated to re-construct the 16-bit wise word. The 16-bit word-wise operation is executed at once with the 16-bit instruction to improve the performance for the 16-bit MSP430 microcontroller. This approach also optimizes the number of required registers, memory accesses, and operations to half numbers rather than 8-bit word wise implementations. For the ARM Cortex-M3 microcontroller, the 8×32 look-up table based ARIA block cipher implementation is further optimized with the novel memory access. The memory access is finely scheduled to fully utilize the 3-stage pipeline architecture of ARM Cortex-M3 microcontrollers. Furthermore, the counter (CTR) mode of operation is more optimized through pre-computation techniques than the electronic code book (ECB) mode of operation. Finally, proposed ARIA implementations on both low-end target microcontrollers (MSP430 and ARM Cortex-M3) achieved (209 and 96 for 128-bit security level, respectively), (241 and 111 for 192-bit security level, respectively), and (274 and 126 for 256-bit security level, respectively). Compared with previous works, the running timing on low-end target microcontrollers (MSP430 and ARM Cortex-M3) is improved by (92.20% and 10.09% for 128-bit security level, respectively), (92.26% and 10.87% for 192-bit security level, respectively), and (92.28% and 10.62% for 256-bit security level, respectively). The proposed ARIA–CTR implementation improved the performance by 6.6% and 4.0% compared to the proposed ARIA–ECB implementations for MSP430 and ARM Cortex-M3 microcontrollers, respectively.

Keywords: ARIA; block cipher; software implementation; counter mode of operation; microcontroller

Citation: Seo, H.; Kim, H.; Jang, K.; Kwon, H.; Sim, M.; Song, G.; Uhm, S. Compact Implementation of ARIA on 16-Bit MSP430 and 32-Bit ARM Cortex-M3 Microcontrollers. *Electronics* **2021**, *10*, 908. https://doi.org/10.3390/electronics10080908

Academic Editor: Juan M. Corchado

Received: 22 February 2021
Accepted: 8 April 2021
Published: 11 April 2021

1. Introduction

The data encryption is important for the network security. The computation of secure encryption requires high overheads for low-end microcontrollers. In order to achieve high availability on low-end microcontrollers, the efficient implementation of block cipher has been actively studied. For the efficient implementation, unique features of target block ciphers should be considered for optimizations. In this paper, we optimized the electronic code book (ECB) and counter (CTR) modes of operation for ARIA block cipher on both MSP430 and Advanced RISC Machine (ARM) Cortex-M3 microcontrollers.

Proposed in 2004, ARIA block cipher [1] is the standards of South Korean and IETF. Recently, the ARIA on low-end Alf and Vegard's RISC (AVR) was presented by [2]. ARIA implementations on 8-bit AVR required 198.3 (for 128-bit security level), 228.0 (for 192-bit security level), and 257.8 (for 256-bit security level) clock cycles per byte, respectively.

However, optimized implementations of ARIA block cipher on both MSP430 and ARM Cortex-M3 microcontrollers have not been studied. Compared with 8-bit AVR microcontrollers, target microcontrollers have different architectures, in terms of word size, instruction set, general purpose registers, and pipeline stages. For this reason, specialized optimization techniques should be investigated for high performance on both MSP430 and ARM Cortex-M3 microcontrollers. In this work, we improved ARIA block cipher on both MSP430 and ARM Cortex-M3 microcontrollers. ARIA implementations are optimized by considering unique features of target microcontrollers and adopting the state-of-art engineering technique. Furthermore, we proposed ARIA–CTR implementations on both target microcontrollers.

Contribution

The first ARIA block cipher on both MSP430 and ARM Cortex-M3 microcontrollers: Primitive operations of ARIA block cipher, such as substitute and diffusion layers, are efficiently optimized for both MSP430 and ARM Cortex-M3 microcontrollers. With these optimized operations, high-speed implementations of ARIA block cipher are achieved.

Optimized ARIA block cipher implementations for 16-bit MSP430 microcontrollers: Two bytes of input data are concatenated to re-construct the 16-bit word. The operation on the 16-bit word is executed at once to improve the performance and reduce the number of required general purpose registers, memory accesses, and operations, for the 16-bit MSP430 microcontroller. Proposed ARIA implementations on 16-bit MSP430 microcontrollers achieved 209 (for 128-bit security level), 241 (for 192-bit security level), and 274 (for 256-bit security level) clock cycles per byte, respectively. Compared with former works on the identical processor, the running timing is optimized by 92.20% (for 128-bit security level), 92.26% (for 192-bit security level), and 92.28% (for 256-bit security level), respectively [1].

Optimized ARIA implementations for ARM Cortex-M3 microcontrollers: For the ARM Cortex-M3 microcontroller, the pre-computed table based ARIA implementation is further optimized. The memory access is finely re-scheduled to utilize the 3-stage pipeline architecture of ARM Cortex-M3 microcontroller. Finally, proposed ARIA implementations on ARM Cortex-M3 microcontrollers achieved 96 (for 128-bit security level), 111 (for 192-bit security level), and 126 (for 256-bit security level) clock cycles per byte, respectively. Compared with former ARIA implementations on the identical processor, the execution timing is enhanced by 10.09%, 10.87%, and 10.62% for 128-bit, 192-bit, and 256-bit security levels, respectively [1].

Efficient implementation of ARIA–CTR on MSP430 and ARM Cortex-M3 microcontrollers: The implementation of ARIA–CTR is further optimized for MSP430 and ARM Cortex-M3 microcontrollers. For the 16-bit MSP430 microcontroller, 1 substitution layer, 1 diffusion layer, and 2 add-round-key operations are optimized away. For the 32-bit ARM Cortex-M3 microcontroller, both $M \cdot S$ layer and M_1 layer are optimized with pre-computation. With the above optimizations, the performance of the proposed ARIA–CTR implementations are improved over the proposed ARIA–ECB implementations by 6.6% and 4.0% for MSP430 and ARM Cortex-M3 microcontrollers, respectively.

The remainder of this paper is organized as follows. Section 2 presents an overview of the ARIA block cipher and previous block cipher implementations on both 16-bit MSP430 and 32-bit ARM Cortex-M3 microcontrollers. In Section 3, proposed implementations of ARIA block cipher on both 16-bit MSP430 and 32-bit ARM Cortex-M3 microcontrollers are presented. In Section 4, the performance evaluation of proposed implementations is described. Finally, the conclusion is given in Section 5.

2. Related Works

2.1. Target Block Cipher: ARIA

ARIA block cipher consists of a substitution layer, diffusion layer, and add-round-key. Similar to AES block cipher, the substitution layer executes an affine transformation of the inversion function on Galois Field and the diffusion layer executes a simple linear map operation. The add-round-key executes eXclusive-OR operation with plaintext and

round key. ARIA encryption and decryption operations share the identical architecture. This feature optimizes the chip size and code size for hardware and software implementations, respectively.

2.2. Target Microcontrollers: 16-Bit MSP430 and 32-Bit ARM Cortex-M3

The MSP430 microcontroller is a representative 16-bit embedded processor board with a clock frequency of 8–16 MHz, 32–48 KB of flash memory, 10 KB of RAM, and 12 general purpose registers from `R4` to `R15`. The microcontroller provides sufficient basic arithmetic instructions for implementations. Instructions for block cipher implementations on the MSP430 microcontroller are described in Table 1.

ARM Cortex-M3 is 32-bit microcontroller and designed for embedded computing services. The microcontroller provides low energy consumption with high performance. Arithmetic instructions take one clock cycle but memory access instructions take more clock cycles. The microcontroller supports the barrel-shifter, which performs rotated or shifted registers without additional costs. Instructions for block cipher implementations on the ARM Cortex-M3 microcontroller are described in Table 2.

Table 1. Instruction set summary of ARX operations on the 16-bit MSP430 microcontroller, where `c` represents carry bit.

asm	Operands	Description	Operation	#Clock
ADD	A, B	Add without Carry	$B \leftarrow A + B$	1
XOR	A, B	Exclusive OR	$B \leftarrow A \oplus B$	1
RLA	A	Logical Shift Left	$c\|A \leftarrow A \ll 1$	1
RLC	A	Rotate Left through Carry	$c\|A \leftarrow A \ll 1\|\|c$	1

Table 2. Instruction set summary of ARX operations on the 32-bit Advanced RISC Machine (ARM) Cortex-M3 microcontroller.

asm	Operands	Description	Operation	#Clock
ADD	C, A, B	Add word without Carry	$C \leftarrow A + B$	1
EOR	C, A, B	Exclusive OR	$C \leftarrow A \oplus B$	1
LSL	C, A, B	Shift Left	$C \leftarrow A \ll B$	1
ROR	C, A, B	Rotate Right	$C \leftarrow A \gg >B$	1

2.3. Former Symmetric Key Cryptography on 16-Bit MSP and 32-Bit ARM Microcontrollers

In [3], an optimized implementation of authenticated encryption on MSP430X microcontrollers was presented. In [4], efficient implementations of AES (132 cycles/byte) and SPECK (103 cycles/byte) block ciphers on the MSP430 microcontroller were presented, respectively. In [5], the encryption mode of the tweakable block cipher of the SCREAM authenticated cipher is implemented in the MSP430 microcontroller. In [6], the implementation of Simeck on the MSP430 microcontroller reduces the code size by 19.32% and improves the execution timing by 3.75 times. In [7], a compact implementation of Chaskey on the MSP430 microcontroller was presented. Similarly, many block ciphers were implemented on MSP430 microcontrollers [8–10].

For the case of ARM processors, many implementations were also investigated. In WISA'13, LEA block cipher on the 32-bit ARM processor was introduced [11]. Primitive operations of LEA block cipher were optimized for the 32-bit ARM microcontroller. In [12], AES–CTR implementations were presented and achieved optimal AES implementations. In [13], the new efficient software design of PRESENT block cipher was presented. The CTR mode of operation takes 2100 cycles on the Cortex-M3 microcontroller, which improves the performance by a factor of 8. In [14], a 384-bit permutation design (i.e., Gimli) is efficiently implemented on the 32-bit ARM processor. In [15], constant time implementations of GIFT

block cipher on the ARM Cortex-M3 microcontroller were presented. The 128-bit data can be encrypted with only about 800 cycles for GIFT-64 and about 1300 cycles for GIFT-128. In [16], fixslicing-based AES implementations were also evaluated on ARM Cortex-M. Similarly, many block ciphers were implemented on ARM Cortex-M microcontrollers [8,17–19].

However, previous works do not optimize the ARIA block cipher on both target microcontrollers (MSP430 and ARM Cortex-M3). In this paper, we present the first optimized implementation of ARIA block cipher on both microcontrollers.

3. Proposed Method

Since the length of the original word of the ARIA block cipher is 8-bit, the implementation of ARIA is efficient for the 8-bit architecture as described in [2]. However, the 8-bit word-based ARIA architecture is not efficient for 16-bit cases. For this reason, optimizations for 16-bit architecture should be considered. For the case of 32-bit, the developer of ARIA block cipher suggested techniques to combine substitute and diffusion layers. This approach efficiently performs the computation with 8×32 look-up table access, which is the optimal method for the 32-bit architecture.

We implemented the ARIA for both MSP430 and ARM Cortex-M3 microcontrollers. For the case of MSP430 microcontrollers, two 8-bit wise operations are combined to construct the 16-bit word for the efficient diffusion layer. The 8-bit wise memory access is efficiently handled for the substitution layer. For the case of ARM Cortex-M3 microcontrollers, the previous look-up table based access is further optimized by considering the 3-stage pipe-lining of the 32-bit ARM Cortex-M3 microcontroller. Particularly, instructions are re-scheduled to avoid pipeline stalls. Byte wise rotation operations are also efficiently implemented with ARM native instruction sets.

3.1. Optimized ARIA Implementation on 16-Bit MSP430

The MSP430 microcontroller has twelve 16-bit registers for general purposes. In Table 3, the general purpose register utilization for ARIA encryption on 16-bit MSP430 microcontrollers is presented. In particular, general purpose registers are used for different purposes, such as plaintext pointer, round key pointer, plaintext, loop counter, and temporal variable.

Table 3. Register utilization for ARIA encryption on the 16-bit MSP430 microcontroller.

Register	Utilization
R4–R11	plaintext #1–#8
R12	plaintext pointer/loop counter/temporal variable #1
R13	round key pointer
R14–R15	temporal variable #2–#3

Diffusion Layer: The diffusion layer executes consecutive XOR operations with 8-bit words in a certain order. Some XOR operations of diffusion layer are repeated several times (T value of Algorithm 1). These repeated parts can be computed once. Then, these results can be used several times through the caching to reduce the number of computations. Detailed descriptions for sequential diffusion layer are presented in Algorithm 1. In Steps 1, 6, 11, and 16, some parts of XOR operations are pre-computed. Then, these results are used several times in following steps (2–5, 7–10, 12–15, and 17–20). However, the target microcontroller only supports 16-bit word size and instructions. The straight-forward implementation of 8-bit wise pre-computation technique (i.e., Algorithm 1) is inefficient for the 16-bit MSP430 microcontroller, because only half of register is utilized during the computation.

Algorithm 1 Sequential diffusion layer of ARIA block cipher for 8-bit AVR [1].

Input: 128-bit input (i[0–15])
Output: 128-bit output (o[0–15])

1: $T \leftarrow i[3] \oplus i[4] \oplus i[9] \oplus i[14]$
2: $o[0] \leftarrow i[6] \oplus i[8] \oplus i[13] \oplus T$
3: $o[5] \leftarrow i[1] \oplus i[10] \oplus i[15] \oplus T$
4: $o[11] \leftarrow i[2] \oplus i[7] \oplus i[12] \oplus T$
5: $o[14] \leftarrow i[0] \oplus i[5] \oplus i[11] \oplus T$
6: $T \leftarrow i[2] \oplus i[5] \oplus i[8] \oplus i[15]$
7: $o[1] \leftarrow i[7] \oplus i[9] \oplus i[12] \oplus T$
8: $o[4] \leftarrow i[0] \oplus i[11] \oplus i[14] \oplus T$
9: $o[10] \leftarrow i[3] \oplus i[6] \oplus i[13] \oplus T$
10: $o[15] \leftarrow i[1] \oplus i[4] \oplus i[10] \oplus T$
11: $T \leftarrow i[1] \oplus i[6] \oplus i[11] \oplus i[12]$
12: $o[2] \leftarrow i[4] \oplus i[10] \oplus i[15] \oplus T$
13: $o[7] \leftarrow i[3] \oplus i[8] \oplus i[13] \oplus T$
14: $o[9] \leftarrow i[0] \oplus i[5] \oplus i[14] \oplus T$
15: $o[12] \leftarrow i[2] \oplus i[7] \oplus i[9] \oplus T$
16: $T \leftarrow i[0] \oplus i[7] \oplus i[10] \oplus i[13]$
17: $o[3] \leftarrow i[5] \oplus i[11] \oplus i[14] \oplus T$
18: $o[6] \leftarrow i[2] \oplus i[9] \oplus i[12] \oplus T$
19: $o[8] \leftarrow i[1] \oplus i[4] \oplus i[15] \oplus T$
20: $o[13] \leftarrow i[3] \oplus i[6] \oplus i[8] \oplus T$

In Algorithm 2, the implementation of a 2-way diffusion layer to utilize the 16-bit word for the 16-bit MSP430 microcontroller is presented. Unlike the straight-forward implementation, two 8-bit words are concatenated to form a 16-bit word (i.e., size of MSP430 microcontroller) and a 16-bit wise XOR operation is performed at once. Similar to the previous approach, the 16-bit wise pre-computation ($TH \| TL$) is performed and the result is utilized by several times in other steps. Compared with the previous approach, the approach halves the number of required number of XOR operations and general purpose registers.

Substitution Layer: The substitution layer can be implemented with the 8×8 look-up table access (i.e., memory access). The 16-bit MSP430 microcontroller supports both word-wise and byte-wise memory access (`.B`). Since the look-up table is 8-bit wise, we utilized byte-wise memory access. In particular, the 16-bit result is accessed twice by 8-bit wise. Detailed procedures for substitution layer on the 16-bit MSP430 microcontroller are given in Figure 1 and described as follows:

- The lower part of the general purpose register (1–8) is used for index of look-up table access. To extract the lower part (8-bit) from the word of the target architecture (16-bit), `MOV.B` instruction is utilized.
- The higher part of the general purpose register (9–16) is used for index of look-up table access. To extract the higher part (8-bit) from the word (16-bit), the higher part and lower part are swapped (`SWPB`) and utilized. Then, the lower part is moved with the `MOV.B` instruction.

Algorithm 2 The 2-way diffusion layer of ARIA block cipher for the 16-bit MSP430 microcontroller.

Input: 128-bit input (i[0–15])
Output: 128-bit output (o[0–15])

1: $\{TH \| TL\} \leftarrow \{i[3] \| i[2]\} \oplus \{i[4] \| i[5]\} \oplus \{i[9] \| i[8]\} \oplus \{i[14] \| i[15]\}$
2: $\{o[0] \| o[1]\} \leftarrow \{i[6] \| i[7]\} \oplus \{i[8] \| i[9]\} \oplus \{i[13] \| i[12]\} \oplus \{TH \| TL\}$
3: $\{o[5] \| o[4]\} \leftarrow \{i[1] \| i[0]\} \oplus \{i[10] \| i[11]\} \oplus \{i[15] \| i[14]\} \oplus \{TH \| TL\}$
4: $\{o[11] \| o[10]\} \leftarrow \{i[2] \| i[3]\} \oplus \{i[7] \| i[6]\} \oplus \{i[12] \| i[13]\} \oplus \{TH \| TL\}$
5: $\{o[14] \| o[15]\} \leftarrow \{i[0] \| i[1]\} \oplus \{i[5] \| i[4]\} \oplus \{i[11] \| i[10]\} \oplus \{TH \| TL\}$
6: $\{TH \| TL\} \leftarrow \{i[1] \| i[0]\} \oplus \{i[6] \| i[7]\} \oplus \{i[11] \| i[10]\} \oplus \{i[12] \| i[13]\}$
7: $\{o[2] \| o[3]\} \leftarrow \{i[4] \| i[5]\} \oplus \{i[10] \| i[11]\} \oplus \{i[15] \| i[14]\} \oplus \{TH \| TL\}$
8: $\{o[7] \| o[6]\} \leftarrow \{i[3] \| i[2]\} \oplus \{i[8] \| i[9]\} \oplus \{i[13] \| i[12]\} \oplus \{TH \| TL\}$
9: $\{o[9] \| o[8]\} \leftarrow \{i[0] \| i[1]\} \oplus \{i[5] \| i[4]\} \oplus \{i[14] \| i[15]\} \oplus \{TH \| TL\}$
10: $\{o[12] \| o[13]\} \leftarrow \{i[2] \| i[3]\} \oplus \{i[7] \| i[6]\} \oplus \{i[9] \| i[8]\} \oplus \{TH \| TL\}$

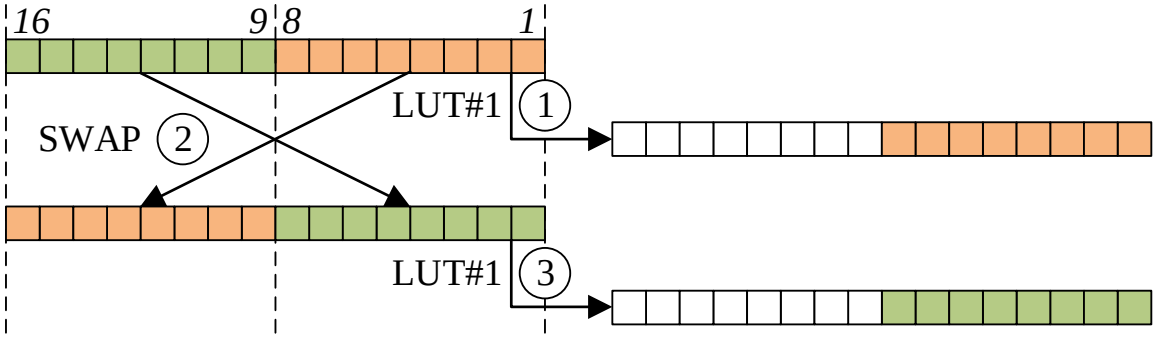

Figure 1. Look-up table access for substitution layer on 16-bit MSP430. Each square block represents 1-bit. ①: LUT access with 1–8-th bits, ②: exchanging lower and higher bytes, ③: LUT access with 9–16-th bits.

Optimization of Counter Mode of Operation for 16-bit Architecture: The counter mode of operation can be skipped through pre-computation with constant variables [2]. Previous works have been devoted to improve the performance of counter mode through the pre-computation [2,20–24]. The input of counter mode of operation consists of counter (32-bit) and constant nonce (96-bit). One substitution and one diffusion, and two add-round-key operations for the 96-bit constant nonce part can be pre-computed. Only the remaining part for the 32-bit counter is computed online. The optimized ARIA–CTR implementation was presented by [2].

In Algorithm 3, the 2-way diffusion layer after the pre-computation is given. In Steps 1–2, computations on counter value are performed. In Steps 3–5, 3 XOR operations are performed with $\{T[13] \parallel T[12]\}$ in 2-way. In Steps 6, the 16-bit word is swapped in byte-wise. Then, 4 XOR operations are performed with $\{T[12] \parallel T[13]\}$. In Steps 11–13, 3 XOR operations are performed with $\{T[15] \parallel T[14]\}$ in the 2-way parallel way. In Step 14, the 16-bit word is swapped in byte-wise. Then, 4 XOR operations are performed with $\{T[14] \parallel T[15]\}$.

Algorithm 3 The 2-way diffusion layer of ARIA block cipher for CTR mode on the 16-bit MSP430 microcontroller.

Input: 128-bit pre-computed (p[0–15]), 32-bit counter (c[0–3])
Output: 128-bit output (o[0–15])
// computation with counter value
1: $\{T[13] \parallel T[12]\} \leftarrow \{S_2[RK_1[13] \oplus c[1]] \| S_1[RK_1[12] \oplus c[0]]\}$
2: $\{T[15] \parallel T[14]\} \leftarrow \{S_2^{-1}[RK_1[15] \oplus c[3]] \| S_1^{-1}[RK_1[14] \oplus c[2]]\}$
3: $\{p[3] \parallel p[2]\} \leftarrow \{p[3] \parallel p[2]\} \oplus \{T[13] \parallel T[12]\}$
4: $\{p[7] \parallel p[6]\} \leftarrow \{p[7] \parallel p[6]\} \oplus \{T[13] \parallel T[12]\}$
5: $\{p[13] \parallel p[12]\} \leftarrow \{p[13] \parallel p[12]\} \oplus \{T[13] \parallel T[12]\}$
// byte-wise swap operation
6: $\{T[12] \parallel T[13]\} \leftarrow \{T[13] \parallel T[12]\}$
7: $\{p[1] \parallel p[0]\} \leftarrow \{p[1] \parallel p[0]\} \oplus \{T[12] \parallel T[13]\}$
8: $\{p[7] \parallel p[6]\} \leftarrow \{p[7] \parallel p[6]\} \oplus \{T[12] \parallel T[13]\}$
9: $\{p[9] \parallel p[8]\} \leftarrow \{p[9] \parallel p[8]\} \oplus \{T[12] \parallel T[13]\}$
10: $\{p[11] \parallel p[10]\} \leftarrow \{p[11] \parallel p[10]\} \oplus \{T[12] \parallel T[13]\}$
11: $\{p[1] \parallel p[0]\} \leftarrow \{p[1] \parallel p[0]\} \oplus \{T[15] \parallel T[14]\}$
12: $\{p[5] \parallel p[4]\} \leftarrow \{p[5] \parallel p[4]\} \oplus \{T[15] \parallel T[14]\}$
13: $\{p[15] \parallel p[14]\} \leftarrow \{p[15] \parallel p[14]\} \oplus \{T[15] \parallel T[14]\}$
// byte-wise swap operation
14: $\{T[14] \parallel T[15]\} \leftarrow \{T[15] \parallel T[14]\}$
15: $\{p[3] \parallel p[2]\} \leftarrow \{p[3] \parallel p[2]\} \oplus \{T[14] \parallel T[15]\}$
16: $\{p[5] \parallel p[4]\} \leftarrow \{p[5] \parallel p[4]\} \oplus \{T[14] \parallel T[15]\}$
17: $\{p[9] \parallel p[8]\} \leftarrow \{p[9] \parallel p[8]\} \oplus \{T[14] \parallel T[15]\}$
18: $\{p[11] \parallel p[10]\} \leftarrow \{p[11] \parallel p[10]\} \oplus \{T[14] \parallel T[15]\}$

3.2. Optimized ARIA Block Cipher on Cortex-M3

ARM Cortex-M3 microcontrollers have 14 32-bit general purpose registers. In Table 4, the register utilization for ARIA encryption on target microcontrollers is presented. Plaintext pointer, round key pointer, look-up table pointer, temporal variables, and plaintext are allocated in registers.

Diffusion and Substitution Layers: In [1], the 8×32 look-up table-based round implementation was presented. The look-up table combines both diffusion and substitution layers for the 32-bit architecture. The diffusion layer A is constructed in the form of $M_1 \cdot M_2 \cdot M_1$ where

$$M_1 = \begin{pmatrix} I & I & I & 0 \\ I & 0 & I & I \\ I & I & 0 & I \\ 0 & I & I & I \end{pmatrix}, \; M_2 = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 \\ 0 & 0 & 0 & P_3 \end{pmatrix} \cdot \begin{pmatrix} T & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{pmatrix},$$

$$T = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \; P_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \; P_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \; P_3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

For simplifying above notations, the following notations are used.

$$P = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 \\ 0 & 0 & 0 & P_3 \end{pmatrix}, \; M = \begin{pmatrix} T & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{pmatrix}.$$

When S is the substitution layer, the round without key addition is performed as follows:

$$A \cdot S = M_1 \cdot M_2 \cdot M_1 \cdot S = M_1 \cdot P \cdot M_1 \cdot M \cdot S.$$

$M \cdot S$ is performed by using 8×32 look-up tables, where M is a block diagonal matrix. As described above, the efficient implementation of each matrix (M_1, P, $M \cdot S$) is important. The optimal implementation is highly related with compact memory access on the target microcontroller. In this paper, we presented the pipelined LUT access method.

Optimization of $M \cdot S$ matrix: The 8×32 table look-up is performed with the 8-bit wise offset. Since the word size of the ARM Cortex-M3 processor is 32-bit long, four 8-bit wise look-up accesses are required for full 32-bit computations. Detailed descriptions are presented in Figure 2. To extract the 8-bit value out of 32-bit, barrel-shifter, rotation, and masking operations are performed. The sequential pre-computed table-based approach performs four pre-computed table accesses, consecutively. However, the read-and-write dependency between source and destination addresses leads to pipeline stalls in this approach and pipeline stalls introduce the timing delay.

To resolve this performance penalty, the pipelined LUT access for $M \cdot S$ layer is proposed in Algorithm 4. The dependency between source and destination addresses is removed by re-alignment of instruction sets. The operation consists of three steps. In Steps 1–6, the offset setting for memory address pointer is performed. This step generates four 8-bit offsets from 32-bit word for four memory address pointers. In Steps 7–10, four memory accesses are performed with four base address pointers, consecutively. In Steps 11–13, results of LUTs are accumulated together. Finally, the result (`Y0`) is returned in Step 14.

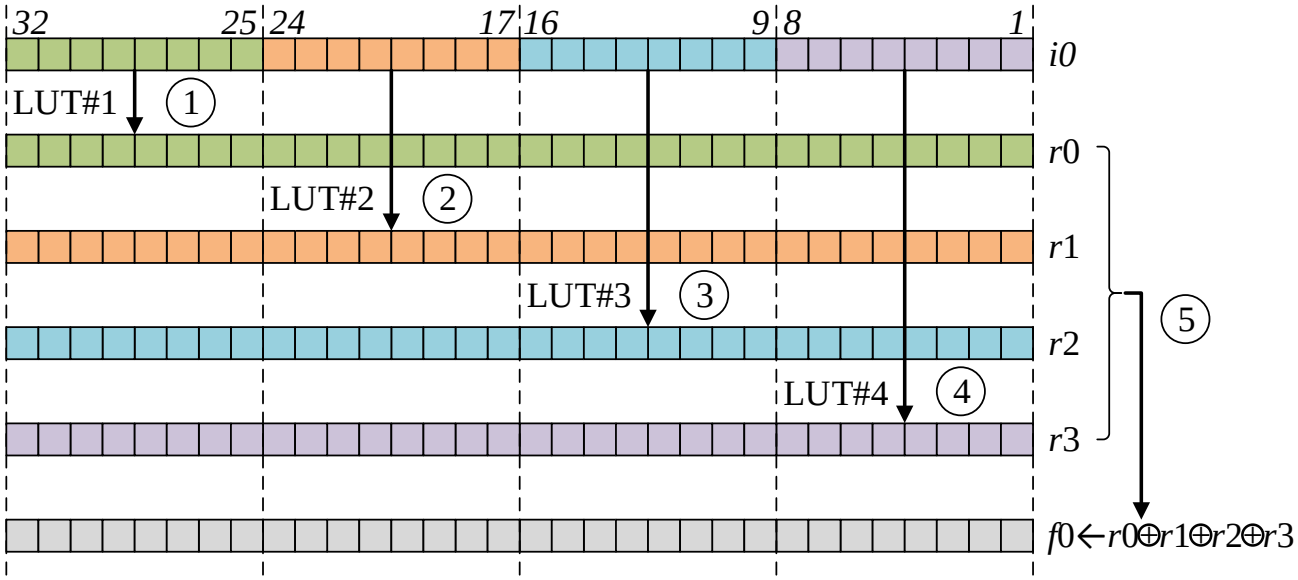

Figure 2. The 8 × 32 table look-ups for ARIA block cipher on ARM Cortex-M3. ①: LUT access with 25–32-th bits, ②: LUT access with 17–24-th bits, ③: LUT access with 9–16-th bits, ④: LUT access with 1–8-th bits, ⑤: XOR operations with LUT results.

Table 4. Register allocation for ARIA encryption on ARM Cortex-M3 processors.

Register	Allocation
R0	plaintext pointer → look-up table pointer #1
R1	round key pointer
R2	look-up table pointer #2
R3–R6	plaintext #1–#4
R7–R11	temporal variables #1–#5
R12	look-up table pointer #3
R14	look-up table pointer #4

In Figure 3, the comparison of computation order between previous and proposed methods is presented. The previous method does not take advantage of pipelining features, while the proposed method achieved the pipelining feature by re-ordering operations. The proposed approach ensures low latency by avoiding the pipeline stall.

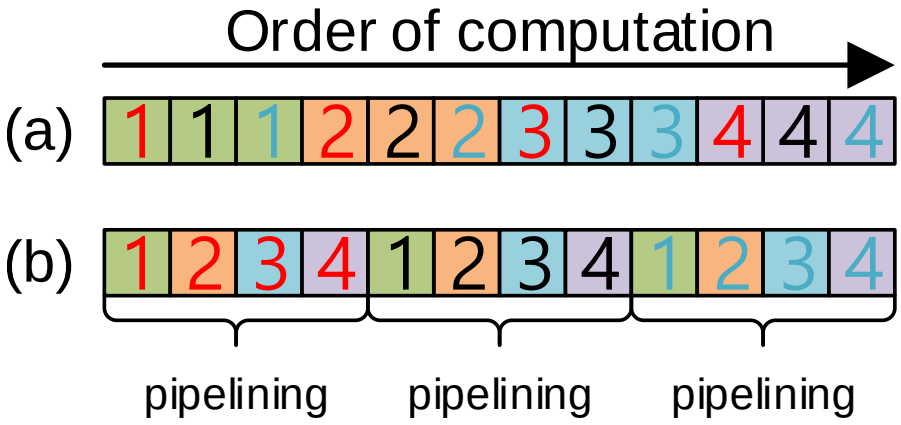

Figure 3. Order of computation between previous and proposed look-up table access. (**a**) Previous approach. (**b**) Proposed approach. Red, black, and blue characters represent offset setting for address pointer, memory access, and result accumulation, respectively. The block indicates the specific operation.

Algorithm 4 Pipelined LUT access for $M \cdot S$ layer of ARIA block cipher on the ARM Cortex-M3 microcontroller.

Input: LUT input X0, LUT memory addresses (P0, P1, P2, and P3)
Output: LUT result Y0
// offset setting for address pointer
1: LSR Y0, X0, #24 {25-32-th bits}
2: AND TMP0, X0, #0XFF0000 {17-24-th bits}
3: ADD TMP0, P1, TMP0, LSR #14
4: AND TMP1, X0, #0XFF00 {9-16-th bits}
5: ADD TMP1, P2, TMP1, LSR #6
6: AND X0, X0, #0XFF {1-8-th bits}
// memory access
7: LDR Y0, [P0, Y0, LSL #2] {LUT#1 access}
8: LDR TMP0, [TMP0] {LUT#2 access}
9: LDR TMP1, [TMP1] {LUT#3 access}
10: LDR X0, [P3, X0, LSL #2] {LUT#4 access}
//result accumulation
11: EOR Y0, Y0, TMP0 $\{r0 \oplus r1\}$
12: EOR Y0, Y0, TMP1 $\{r0 \oplus r1 \oplus r2\}$
13: EOR Y0, Y0, X0 $\{r0 \oplus r1 \oplus r2 \oplus r3\}$
14: **return** Y0

Optimization of M_1 matrix: The implementation of M_1 layer consists of 6 XOR operations. Descriptions are presented in Algorithm 5. The M_1 matrix is performed twice in each round.

Algorithm 5 M_1 layer of ARIA block cipher on ARM Cortex-M3.

Input: Intermediate result (T0, T1, T2, T3)
Output: Result (T0, T1, T2, T3)
1: EOR T1, T1, T2
2: EOR T2, T2, T3
3: EOR T0, T0, T1
4: EOR T3, T3, T1
5: EOR T2, T2, T0
6: EOR T1, T1, T2

Optimization of P matrix: The P layer performs three byte-wise rotation operations. These rotation operations are efficiently performed with dedicated instructions of target ARM processor. Detailed descriptions of P layer of ARIA block cipher on the ARM Cortex-M3 microcontroller are shown in Algorithm 6.

Algorithm 6 P layer of ARIA block cipher on the ARM Cortex-M3 microcontroller.

Input: Intermediate result (T1, T2, T3)
Output: Result (X1, X2, X3)
1: REV16 X1, T1 {reverse byte order in each halfword independently}
2: ROR X2, T2, #16 {right rotate by 16-bit}
3: REV X3, T3 {reverse byte order in a word}

Optimization of Counter Mode of Operation for 32-bit Architecture: Previous optimization methods for counter mode of operation are not available in the ARM Cortex-M3 microcontroller [2], since 32-bit ARM Cortex-M3 implementation employed the LUT method while the previous approach utilized the 8-bit S-box-based implementation. For that reason, the CTR technique is re-designed for the LUT-based implementation. First, $M \cdot S$ layer is optimized. Only the 32-bit counter part is calculated online for this layer. Second, M_1 layer is also optimized. Only the computation with 32-bit counter part is computed. The detailed M_1 layer is given in Algorithm 7. Only three XOR operations are performed.

Algorithm 7 Optimized M_1 layer of ARIA block cipher for counter mode of operation on ARM Cortex-M3.

Input: Intermediate result (`T1, T2, T3`)
Output: Result (`T1, T2, T3`)

```
1: EOR T2, T2, T3
2: EOR T3, T3, T1
3: EOR T1, T1, T2
```

3.3. Secure Implementation of ARIA

Software implementations of block cipher should be secure against the side-channel attack. The proposed ARIA implementation is secure against the most popular and effective attack (i.e., timing attack) on software implementations (https://www.bearssl.org/constanttime.html, accessed date: 10 April 2021) [25]. In order to avoid the timing attack, proposed implementations do not include conditional branch statements depending on the secret information. Regardless of the secret key, the implementation always executes same operations and this ensures the constant timing of implementations. Furthermore, since the target embedded processor does not provide the cache memory, the memory access pattern is always the regular fashion. The attacker cannot exploit the cache timing attack on this case and the implementation is secure against the timing attack.

4. Evaluation

We evaluated optimized ARIA implementations on both MSP430 (MSP430F1611) and ARM microcontrollers (Arduino DUE). Comparison results in terms of RAM (bytes), program code size (bytes), and execution timing (clock cycles) are presented in Tables 5 and 6 for MSP430 and ARM Cortex-M3, respectively. The proposed implementation is the first ARIA optimization on both MSP430 and ARM Cortex-M3 microcontrollers. The comparison is performed with previous implementations in [1]. 16-bit MSP430 implementations utilized the 8-bit pre-computation result in ROM (Storing results into the RAM is also possible but the target processor has limited size of the RAM. For this reason, we only consider the ROM). The utilization of code and RAM are similar to the previous implementation. However, the execution timing is significantly improved by 92.2% compared to the previous work. The performance improvement is mainly coming from the 2-way computation (i.e., 16-bit wise) of diffusion layer and optimized memory access. Proposed ARIA–CTR implementations show better performance than proposed ARIA–ECB implementations by 6.6% (for 128-bit security level), 5.3% (for 192-bit security level), and 5.1% (for 256-bit security level), respectively.

For the ARM microcontroller, the look-up table is stored in different storage types (i.e., ROM and RAM). The RAM/ROM-based implementation improved the execution timing by 10.09/14.13% (for 128-bit security level), 10.87/15.12% (for 192-bit security level), and 10.62/14.42% (for 256-bit security level), compared to previous implementations, respectively. The utilization of RAM is similar to previous implementations. The code size of proposed implementation is smaller than previous work for the 128-bit ARIA implementation with ROM. For the 192-bit and 256-bit cases for ROM, the code size of proposed work is bigger than the previous work. Proposed RAM-based implementations achieved smaller code size than previous works in all security levels. The RAM based implementation achieved better performance but used more RAM storage than the ROM-based implementation. Proposed RAM/ROM based ARIA–CTR implementations achieved better performance than proposed ARIA–ECB implementations by 2.04/4.00% (for 128-bit security level), 2.33%/3.47% (for 192-bit security level), and 1.54%/3.05% (for 256-bit security level), respectively.

Table 5. Performance evaluation of ARIA on MSP430 in terms of code size (bytes), RAM (bytes), and execution time (clock cycles per byte), where 8*t*, *o*, and *c* represent 8 × 8 pre-computation-based implementation, pre-computation stored in ROM, and counter mode of operation, respectively. EKS, ENC, and SUM represent encryption key scheduling, encryption, decryption, and summation, respectively.

Implementation	Options			Code Size (Bytes)			RAM (Bytes)		Execution Time (Cycles per Byte)	
	8*t*	***o***	***c***	**EKS**	**ENC**	**SUM**	**EKS**	**ENC**	**EKS**	**ENC**
ARIA-128										
Kwon et al. [1]	√	√	–	2708	1760	2966	288	256	9947	2680
Proposed method	√	√	–	8206	1756	7818	312	248	345	209
Proposed method	√	√	√	–	2872	2872	–	280	–	195
ARIA-192										
Kwon et al. [1]	√	√	–	2708	1760	2966	320	288	7586	3117
Proposed method	√	√	–	8856	1756	8468	352	280	380	241
Proposed method	√	√	√	–	2872	2872	320	–	–	228
ARIA-256										
Kwon et al. [1]	√	√	–	2708	1760	2966	352	320	6404	3551
Proposed method	√	√	–	9556	1756	9168	392	312	209	274
Proposed method	√	√	–	–	2872	2872	360	–	–	260

Table 6. Performance evaluation of ARIA on ARM-M3 in terms of code size (bytes), RAM (bytes), and execution time (clock cycles per byte), where 32*t* *a*, *o*, and *c* represent 8 × 32 pre-computation-based implementation, pre-computation stored in RAM, pre-computation stored in ROM, and counter mode of operation, respectively. EKS, ENC, and SUM represent encryption key scheduling, encryption, decryption, and summation, respectively.

Implementation	Options				Code Size (Bytes)			RAM (Bytes)		Execution Time (Cycles per Byte)	
	32*t*	***a***	***o***	***c***	**EKS**	**ENC**	**SUM**	**EKS**	**ENC**	**EKS**	**ENC**
ARIA-128											
Kwon et al. [1]	√	–	√	–	6504	10,636	11,408	236	224	92	160
Proposed method	√	–	√	–	5872	7688	9336	236	224	67	147
Proposed method	√	–	√	–	–	11,500	11,500	–	296	–	144
Kwon et al. [1]	√	√	–	–	2408	4816	7312	4332	4320	80	112
Proposed method	√	√	–	–	1776	3592	5240	4332	4320	55	100
Proposed method	√	√	–	√	–	3296	3296	–	8500	–	96
ARIA-192											
Kwon et al. [1]	√	–	√	–	6504	10,636	11,408	268	256	98	187
Proposed method	√	–	√	–	8840	8360	12,936	268	256	71	171
Proposed method	√	–	√	√	–	12,300	12,300	–	336	–	167
Kwon et al. [1]	√	√	–	–	2408	4816	7312	4364	4352	86	131
Proposed method	√	√	–	–	4744	4264	5928	4364	4352	58	115
Proposed method	√	√	–	√	–	3864	3864	–	8540	–	111
ARIA-256											
Kwon et al. [1]	√	–	√	–	6504	10,636	11,408	300	288	103	212
Proposed method	√	–	√	–	8840	12,288	12,936	300	288	74	194
Proposed method	√	–	√	√	–	12,300	12,300	–	376	–	191
Kwon et al. [1]	√	√	–	–	2408	4816	7312	4396	4384	92	148
Proposed method	√	√	–	–	4744	8192	6600	4396	4384	61	131
Proposed method	√	√	–	√	–	4416	4416	–	8580	–	127

5. Conclusions

We presented the new compact implementation of ARIA block cipher on microcontrollers, namely MSP430 and ARM Cortex-M3. We firstly optimized the implementation of ARIA block cipher. The 2-way computations of diffusion layer and optimized memory access are presented targeting for the MSP430 microcontroller. Pipelined memory access and optimized byte-wise rotation are presented for the ARM microcontroller. For the 16-bit word diffusion layer, two 8-bit words are combined to construct the 16-bit word and the two 8-bit operations are performed in a single 16-bit operation of the 16-bit MSP430 microcontroller (i.e., parallel approach). For the pipelined memory access, memory offset, memory access, and calculation are finely re-scheduled to meet the 3-stage pipeline, which avoids pipeline stalls in consecutive LUT accesses. Lastly, we proposed the efficient implementation of ARIA–CTR for both embedded processors. This method takes advantages of pre-computation of constant nonce value.

In this paper, we proposed compact ARIA implementations on microcontrollers. With this technique, we can pursue several future works. First, we can utilize the proposed method to the efficient implementation of CTR [2,21,26]. By combining both techniques, we can find further improvements on ARIA implementations for specific purposes. Second, the recent work considered the secure block cipher implementation on ARM Cortex-M4 microcontrollers [19]. We can apply the secure implementation technique to proposed ARIA implementations for high security. Third, we investigated the block cipher implementation of low-end embedded processors. We will study on the block cipher implementation on 64-bit AMD and 32-bit RISC-V processors.

Author Contributions: Investigation, H.S., H.K. (Hyunjun Kim), and H.K. (Hyeokdong Kwon); Software, H.S., K.J. and M.S.; Writing-original draft, H.S., G.S. and S.U.; Writing-review and editing, H.S., H.K. (Hyunjun Kim), K.J., H.K. (Hyeokdong Kwon), M.S., G.S. and S.U. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services) and this work was partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2020R1F1A1048478). This research was financially supported by Hansung University for Hwajeong Seo.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kwon, D.; Kim, J.; Park, S.; Sung, S.H.; Sohn, Y.; Song, J.H.; Yeom, Y.; Yoon, E.J.; Lee, S.; Lee, J.; et al. New block cipher: ARIA. In Proceedings of the International Conference on Information Security and Cryptology, Seoul, Korea, 27–28 November 2003; pp. 432–445.
2. Seo, H.; Kwon, H.; Kim, H.; Park, J. ACE: ARIA-CTR Encryption for Low-End Embedded Processors. *Sensors* **2020**, *20*, 3788. [CrossRef] [PubMed]
3. Gouvêa, C.P.; López, J. High speed implementation of authenticated encryption for the MSP430X microcontroller. In *Progress in Cryptology–LATINCRYPT 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 288–304.
4. Buhrow, B.; Riemer, P.; Shea, M.; Gilbert, B.; Daniel, E. Block cipher speed and energy efficiency records on the MSP430: System design trade-offs for 16-bit embedded applications. In Proceedings of the International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, 2–4 October 2014; pp. 104–123.
5. Diehl, W. Implementation of the SCREAM Tweakable Block Cipher in MSP430 Assembly Language. *IACR Cryptol. ePrint Arch.* **2015**, *2015*, 761.
6. Park, T.; Seo, H.; Lee, G.; Kim, H. Efficient implementation of simeck family block cipher on 16-bit MSP430. In Proceedings of the 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, Italy, 4–7 July 2017; pp. 983–988.
7. Lee, G.; Seo, H.; Park, T.; Kim, H. Optimized implementation of chaskey MAC on 16-bit MSP430. In Proceedings of the 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, Italy, 4–7 July 2017; pp. 904–909.
8. Seo, H.; Jeong, I.; Lee, J.; Kim, W. Compact Implementations of ARX-Based Block Ciphers on IoT Processors. *ACM Trans. Embed. Comput. Syst.* **2018**, *17*, 60. [CrossRef]
9. Seo, H.; An, K.; Kwon, H. Compact LEA and HIGHT implementations on 8-bit AVR and 16-bit MSP processors. In Proceedings of the International Workshop on Information Security Applicationsm, Jeju Island, Korea, 23–25 August 2018; pp. 253–265.

10. Dinu, D.; Le Corre, Y.; Khovratovich, D.; Perrin, L.; Großschädl, J.; Biryukov, A. Triathlon of Lightweight Block Ciphers for the Internet of Things. *IACR Cryptol. ePrint Arch.* **2015**, *2015*, 209. [CrossRef]
11. Hong, D.; Lee, J.; Kim, D.; Kwon, D.; Ryu, K.H.; Lee, D.G. LEA: A 128-bit block cipher for fast encryption on common processors. In Proceedings of the International Workshop on Information Security Applications, Jeju Island, Korea, 19–21 August 2013; pp. 3–27.
12. Schwabe, P.; Stoffelen, K. All the AES you need on Cortex-M3 and M4. In Proceedings of the International Conference on Selected Areas in Cryptography, St. John's, NL, Canada, 10–12 August 2016; pp. 180–194.
13. Reis, T.B.; Aranha, D.F.; López, J. PRESENT runs fast. In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Taipei, Taiwan, 25–28 September 2017; pp. 644–664.
14. Bernstein, D.J.; Kölbl, S.; Lucks, S.; Massolino, P.M.C.; Mendel, F.; Nawaz, K.; Schneider, T.; Schwabe, P.; Standaert, F.X.; Todo, Y.; et al. Gimli: A cross-platform permutation. In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Taipei, Taiwan, 25–28 September 2017; pp. 299–320.
15. Adomnicai, A.; Najm, Z.; Peyrin, T. Fixslicing: A new GIFT representation. *IACR Trans. Cryptogr. Hardw. Embed.* **2020**, *2020*, 402–427.
16. Adomnicai, A.; Peyrin, T. Fixslicing AES-like Ciphers. *IACR Trans. Cryptogr. Hardw. Embed.* **2021**, *2021*, 402–425.
17. Koo, B.; Roh, D.; Kim, H.; Jung, Y.; Lee, D.G.; Kwon, D. CHAM: A Family of Lightweight Block Ciphers for Resource-Constrained Devices. In Proceedings of the International Conference on Information Security and Cryptology, Xi'an, China, 3–5 November 2017; pp. 3–25.
18. Seo, H. High Speed Implementation of LEA on ARM Cortex-M3 processor. *J. Korea Inst. Inf. Commun.* **2018**, *22*, 1133–1138.
19. Seo, H.; Liu, Z. All the HIGHT You Need on Cortex–M4. In Proceedings of the International Conference on Information Security and Cryptology, Nanjing, China, 6–8 December 2019; pp. 70–83.
20. Park, J.H.; Lee, D.H. FACE: Fast AES CTR mode Encryption Techniques based on the Reuse of Repetitive Data. *IACR Trans. Cryptogr. Hardw. Embed.* **2018**, *2018*, 469–499. [CrossRef]
21. Kim, K.; Choi, S.; Kwon, H.; Liu, Z.; Seo, H. FACE–LIGHT: Fast AES–CTR Mode Encryption for Low-End Microcontrollers. In Proceedings of the International Conference on Information Security and Cryptology, Nanjing, China, 6–8 December 2019; pp. 102–114.
22. Kwon, H.; An, S.; Kim, Y.; Kim, H.; Choi, S.J.; Jang, K.; Park, J.; Kim, H.; Seo, S.C.; Seo, H. Designing a CHAM Block Cipher on Low-End Microcontrollers for Internet of Things. *Electronics* **2020**, *9*, 1548. [CrossRef]
23. Kim, Y.; Kwon, H.; An, S.; Seo, H.; Seo, S.C. Efficient implementation of ARX-based block ciphers on 8-Bit AVR microcontrollers. *Mathematics* **2020**, *8*, 1837. [CrossRef]
24. Kwon, H.; Kim, Y.; Seo, S.C.; Seo, H. High-Speed Implementation of PRESENT on AVR Microcontroller. *Mathematics* **2021**, *9*, 374. [CrossRef]
25. Bernstein, D.J. Cache-Timing Attacks on AES. 2005. Available online: https://cr.yp.to/antiforgery/cachetiming-20050414.pdf (accessed on 22 February 2021).
26. Kim, K.; Choi, S.; Kwon, H.; Kim, H.; Liu, Z.; Seo, H. PAGE-Practical AES-GCM Encryption for Low-End Microcontrollers. *Appl. Sci.* **2020**, *10*, 3131. [CrossRef]