Design and Implementation of Unified Hardware for 128-Bit Block Ciphers ARIA and AES

Bonseok Koo, Gwonho Ryu, Taejoo Chang, and Sangjin Lee

ABSTRACT—ARIA and the Advanced Encryption Standard (AES) are next generation standard block cipher algorithms of Korea and the US, respectively. This letter presents an area-efficient unified hardware architecture of ARIA and AES. Both algorithms have 128-bit substitution permutation network (SPN) structures, and their substitution and permutation layers could be efficiently merged. Therefore, we propose a 128-bit processor architecture with resource sharing, which is capable of processing ARIA and AES. This is the first architecture which supports both algorithms. Furthermore, it requires only 19,056 logic gates and encrypts data at 720 Mbps and 1,047 Mbps for ARIA and AES, respectively.

Keywords—ARIA, AES, hardware architecture, resource sharing.

I. Introduction

The Advanced Encryption Standard (AES) is a block cipher which was adopted as an encryption standard by the US government in November 2001 [1]. In Korea, ARIA was announced as a standard block cipher algorithm in December 2004 [2]. The algorithm structures of ARIA and AES are analogous. Both algorithms are 128-bit block ciphers with substitution permutation network (SPN) structures. Also, ARIA uses two kinds of S-boxes, one of which is the same as the S-box of AES. However, the permutation layer of ARIA uses a 16×16 involutional binary matrix, whereas AES uses two kinds of 4×4 matrices which involve multiplication over GF(2^8).

In this letter, using resource sharing techniques, we propose an efficient unified hardware architecture incorporating ARIA and AES, and we measure its performance using a 0.25 μm CMOS standard cell library. Our processor is very suitable to implement low-cost and area-constrained cryptographic modules that should support both algorithms. Note also that the Public Key Cryptography Standard (PKCS) #11 has recently begun to support the mechanisms of ARIA as well as those of AES [3].

Manuscript received Apr. 17, 2007; revised Sept. 3, 2007.

Bonseok Koo (phone: + 82 42 870 2218, email: bskoo@ensec.re.kr), Gwonho Ryu (email: jude@ensec.re.kr), and Taejoo Chang (email: tchang@ensec.re.kr) are with the Network & Communication Security Division, The Attached Institute of ETRI, Daejeon, Rep. of Korea.

Sangjin Lee (email: sangjin@korea.ac.kr) is with the Center for Information Security Technologies, Korea University, Seoul, Rep. of Korea.

II. Unified Substitution Layer

The substitution layer of ARIA consists of two kinds of S-boxes, S_1, S_2, and their inverses. The S-box S_1 is composed of multiplicative inversion over GF(2^8) and affine transformation, while S_2 is a combination of x^{247} and affine transformation. Both S-boxes are defined over the GF(2^8) with the irreducible polynomial $m(x)=x^8+x^4+x^3+x+1$, which is the same as that of AES. Also, S_1 is identical to the S-box of AES. The term x^{247} in S_2 is equivalent to x^{-8} in GF(2^8) and can be represented as $C\cdot x^{-1}$ with an 8×8 binary matrix C. Consequently, the two S-boxes can be rearranged with multiplicative inverse as (1) and (2). Here, A, B, and C are 8×8 binary matrices, and a and b are 8×1 binary vectors:

$$S_1(x) = A\cdot x^{-1} \oplus a,$$
$$S_1^{-1}(x) = (A^{-1}\cdot x \oplus A^{-1}\cdot a\,)^{-1}, \tag{1}$$

$$S_2(x) = B\cdot x^{247} \oplus b = B\cdot x^{-8} \oplus b = BC\cdot x^{-1} \oplus b,$$
$$S_2^{-1}(x) = ((BC)^{-1}\cdot x \oplus (BC)^{-1}\cdot b\,)^{-1}. \tag{2}$$

Therefore, we designed two kinds of shared S-boxes, S_A, S_U, which share a multiplicative inverter and merging affine transformations, as depicted in Fig. 1. Here, S_A executes S_1 or S_1^{-1}, and S_U selectively executes S_1, S_1^{-1}, S_2, or S_2^{-1}. Also, δ_A and δ_U are isomorphism functions from GF(2^8) to GF($((2^2)^2)^2$), and AF_1 and AF_2 denote the affine transformations of S_1 and S_2, respectively.

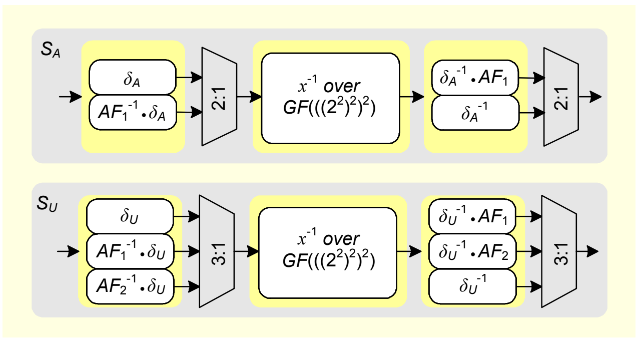

Fig. 1. Structures of the shared S-boxes S_A and S_U.

Table 1. Performance of the shared S-boxes S_A and S_U.

Shared S-box (functions)	Gate counts	Delay (ns)
S_A (S_1, S_1^{-1})	273	6.17
S_U (S_1, S_1^{-1}, S_2, S_2^{-1})	373	6.63

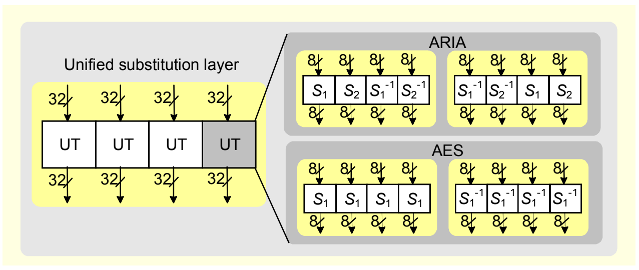

Fig. 2. Structure of the unified substitution layer.

For a compact area, we use the composite field GF$(((2^2)^2)^2)$ inverter and also combine the isomorphism functions and affine transformations as $AF_1^{-1}\cdot\ \delta_A$. The performance of the S-boxes is shown in Table 1, where a 0.25 µm CMOS standard cell library is used, and one-gate is equivalent to a 2-input NAND gate. Although S_U supports four functions of S-boxes, it uses only 373 gates and has a 6.63 ns delay. It is 37% larger and shows a 7% delay as compared to S_A.

Using the shared S-boxes S_A and S_U, we designed a unified transformation (UT), which is equal to (S_A, S_U, S_A, S_U), and finally propose the 128-bit unified substitution layer which consists of four UTs, as depicted in Fig. 2. By switching the selection signals of the multiplexers shown in Fig. 1, UT can operate as (S_1, S_2, S_1^{-1}, S_2^{-1}) or (S_1^{-1}, S_2^{-1}, S_1, S_2) for ARIA and as four units of S_1 or S_1^{-1} for AES.

III. Merging Permutation Functions

AES uses two permutation functions, MixColumns and InvMixColumns in encryption and decryption, respectively. Each function involves addition and multiplication over GF(2^8) and can be written as a 4×4 matrix multiplication [1]. Also, InvMixColumns can be arranged in a form which includes a MixColumns matrix; thus, these two functions can be efficiently implemented with resource sharing [4].

On the other hand, the permutation function of ARIA called a diffusion matrix uses a 16×16 involutional binary matrix [2]. As each row of the matrix has 7 elements of 1, it requires 768 (that is, 6×16×8) 2-input XOR gates to implement the diffusion matrix in a general way.

In order to efficiently design a merged permutation circuit, we examine common 2-input XOR terms among these three matrices. For a 16-byte input ($x_0, x_1, \cdots, x_{14}, x_{15}$), we find 1-byte sharing common terms α_i and β_j as

$$\begin{aligned} \alpha_i &= x_i \oplus x_{4\lfloor i/4 \rfloor+(i+1 \bmod 4)},\ i=0,1,2,\cdots,15, \\ \beta_j &= x_{2j-(j \bmod 2)} \oplus x_{2(j+1)-(j \bmod 2)},\ j=0,1,2,\cdots,7. \end{aligned} \tag{3}$$

Then, using common terms α_i and β_j, each 16-byte output of MixColumns, (u_0, $u_1,\cdots$, u_{14}, u_{15}) and of InvMixColumns, ($v_0, v_1,\cdots, v_{14}, v_{15}$) can be rearranged as

$$\begin{aligned} u_k &= 2\alpha_k \oplus \alpha_{4\lfloor k/4 \rfloor+(k+2 \bmod 4)} \oplus x_{4\lfloor k/4 \rfloor+(k+1 \bmod 4)}, \\ v_k &= u_k \oplus 12\beta_{2\lfloor k/4 \rfloor+(k \bmod 2)} \oplus 8\beta_{2\lfloor k/4 \rfloor+(k+1 \bmod 2)}, \end{aligned} \tag{4}$$

where k=0,$\cdots$,15.

Also, the 16-byte output ($w_0, w_1,\cdots, w_{14}, w_{15}$) of the diffusion matrix of ARIA can be arranged with the common terms as follows:

$$\begin{aligned} w_0 &= x_3 \oplus \beta_2 \oplus \alpha_8 \oplus \alpha_{13}, & w_1 &= x_2 \oplus \beta_3 \oplus \alpha_8 \oplus \alpha_{15}, \\ w_2 &= x_1 \oplus \beta_2 \oplus \alpha_{10} \oplus \alpha_{15}, & w_3 &= x_0 \oplus \beta_3 \oplus \alpha_{10} \oplus \alpha_{13}, \\ w_4 &= x_5 \oplus \beta_0 \oplus \alpha_{11} \oplus \alpha_{14}, & w_5 &= x_4 \oplus \beta_1 \oplus \alpha_9 \oplus \alpha_{14}, \\ w_6 &= x_7 \oplus \beta_0 \oplus \alpha_9 \oplus \alpha_{12}, & w_7 &= x_6 \oplus \beta_1 \oplus \alpha_{11} \oplus \alpha_{12}, \\ w_8 &= x_{10} \oplus \beta_7 \oplus \alpha_0 \oplus \alpha_7, & w_9 &= x_{11} \oplus \beta_6 \oplus \alpha_0 \oplus \alpha_5, \\ w_{10} &= x_8 \oplus \beta_7 \oplus \alpha_2 \oplus \alpha_5, & w_{11} &= x_9 \oplus \beta_6 \oplus \alpha_2 \oplus \alpha_7, \\ w_{12} &= x_{12} \oplus \beta_5 \oplus \alpha_1 \oplus \alpha_6, & w_{13} &= x_{13} \oplus \beta_4 \oplus \alpha_3 \oplus \alpha_6, \\ w_{14} &= x_{14} \oplus \beta_5 \oplus \alpha_3 \oplus \alpha_4, & w_{15} &= x_{15} \oplus \beta_4 \oplus \alpha_1 \oplus \alpha_4. \end{aligned} \tag{5}$$

Because each w_k contains three common terms, one β_j and two different α_i's as in (5), we can save 384 (that is, 3×16×8) 2-input XOR gates. Table 2 shows the hardware size and delay time of permutation functions by sharing common terms. When ARIA diffusion is also implemented, there is about a 49% increase in size from AES functions and no degradation in delay, because v_k makes the critical path.

Table 2. Performance of 16-byte permutation functions.

Permutation functions	no. of XORs	Delay (ns)
MixCol.+InvMixCol.	780	1.64
MixCol.+InvMixCol.+ARIA Diff.	1,164	1.64

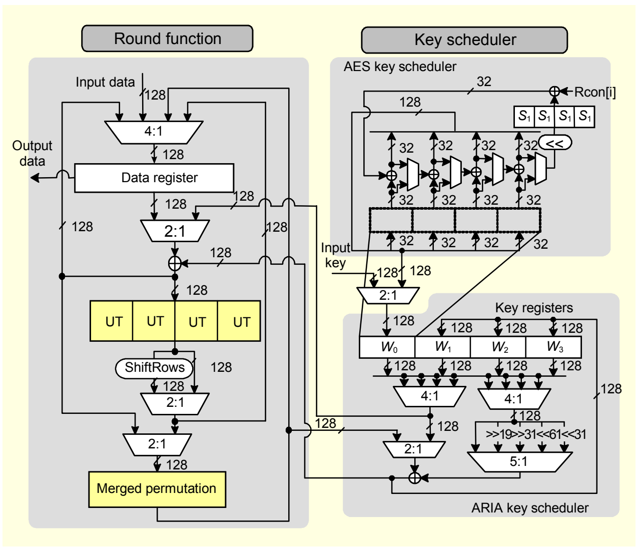

Fig. 3. Architecture of ARIA and AES integrated processor.

IV. Architecture of the Processor

The architecture of the ARIA-AES integrated hardware processor is shown in Fig. 3. The proposed processor is comprised of a round function block and a key scheduler block. Around the unified substitution and merged permutation layers, the round function block is composed of registers for 128-bit data storage, Multiplexers for data path selection, and so on. As the key scheduling processes of ARIA and AES are much different, the key scheduling block is just designed to share a 128-bit register for the round key. That is, the register for the W_0 which is one of the four 128-bit initial keys of ARIA and the register for the round key of AES are shared.

In the AES key scheduling, it is necessary to have four S-boxes performing the S_1 function. We use four separate S_1 S-boxes as shown in Fig. 3; therefore, the round keys of the processor are derived on the fly. As a result, the proposed processor requires 11 clock cycles for AES encryption and 21 cycles for AES decryption. In the case of ARIA, it uses 3 clock cycles for round key initialization, and a total of 16 clock cycles are required to encrypt or decrypt one block.

V. Results and Performance

The proposed processor was implemented with Verilog HDL and was synthesized with a 0.25 μm CMOS standard cell library. Our processor has a compact size of 19,056 gate counts, and its maximum clock frequency is about 90 MHz.

An analysis of comparison with some existing designs is shown in Table 3. The proposed processor is able to perform the encryption or decryption of ARIA and AES and has a 128-bit unit structure. Nevertheless, it is only 53% larger than the smallest AES processor [4] with 128-bit unit architecture. Also, it is only 23% larger than the compact ARIA processor which we designed for comparison. Therefore, the unified processor is about 32% smaller than separate implementations. The processor in [5] supports both AES and Camellia, and is 22% smaller than our processor. However, it requires more than twice as many clock cycles as our processor because it has a 64-bit unit architecture.

Table 3. Comparison of performance with previous results.

Designs (μm)	Gate counts	Functions	Clock cycles	Throughput (Mbps)
Our proposed (0.25)	19,056	ARIA	16	720
		AES (e/d)	11/21	1,047/548
Our ARIA (0.25)	15,496	ARIA	16	816
[4] (0.18)	12,454	AES	11	1,691
[5] (0.18)	14,918	Camellia	22	661
		AES	31	469

VI. Conclusion

In this letter, we proposed a hardware processor with a compact architecture integrating ARIA and AES. We designed an area efficient S-box for both algorithms by using composite field arithmetic on $GF(((2^2)^2)^2)$. We also merged the permutation matrices of the two algorithms into one circuit by extracting their common terms. The proposed processor is the first single hardware circuit which supports both the ARIA and AES algorithms. With 0.25 μm CMOS technology, our processor occupies 19,056 gate counts, which is 32% smaller than discrete implementations, and shows 720 Mbps and 1,047 Mbps for ARIA and AES, respectively.

References

[1] FIPS pub. 197: *Specification for the AES*, Nov. 2001, available at: http://crsc.nist.gov/publications/fips/fips197/fips-197/pdf.

[2] NSRI: *Specification of ARIA*, available at: http://www.nsri.re.kr/ARIA/doc/ARIA-specification-e.pdf.

[3] PKCS#11 v2.20 Amendment 3 Rev. 1: *Additional PKCS#11 Mechanisms*, available at: ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20a3.pdf.

[4] A. Satoh, S. Morioka, K. Takano, and Seiji Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," *ASIACRYPT'01*, Springer-Verlag, 2001, pp. 239-254.

[5] A. Satoh and S. Morioka, "Unified Hardware Architecture for 128-Bit Block Ciphers AES and Camellia," *CHES*, 2003, pp. 304-318.