New Block Cipher: ARIA

Daesung Kwon[1], Jaesung Kim[2], Sangwoo Park[1], Soo Hak Sung[3], Yaekwon Sohn[2], Jung Hwan Song[4], Yongjin Yeom[1], E-Joong Yoon[1], Sangjin Lee[5], Jaewon Lee[2], Seongtaek Chee[1], Daewan Han[1], and Jin Hong[1]

[1] National Security Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon 305-350, KOREA
{ds_kwon, psw, yjyeom, yej, chee, dwh, jinhong}@etri.re.kr
[2] International Science Culture Institute,
P. O. Box 200, Socho-gu 137-602, KOREA
{ijkim1, shrek52, bokmin48}@lycos.co.kr
[3] Department of Computing information & mathematics, Paichai University,
426-6 Doma-dong, Seo-gu, Daejeon 302-735 KOREA
sungsh@mail.pcu.ac.kr
[4] Department of Mathematics, Hanyang University,
17 Haengdang-dong, Seongdong-gu, Seoul 133-791, KOREA
camp123@hanyang.ac.kr
[5] Graduate School of Information Security, Korea University,
1, 5-Ka, Anam-dong, Sungbuk-ku, Seoul 136-701, KOREA
sangjin@korea.ac.kr

Abstract. In this paper, we propose a 128-bit block cipher ARIA which is an involution substitution and permutation encryption network(SPN). We use the same S-boxes as Rijndael to eliminate defects which are caused by a totally involution structure. In the diffusion layer of ARIA, a 16×16 binary matrix of the maximum branch number 8 is used to avoid some attacks well applied to the reduced round of Rijndael. ARIA uses only basic operations, S-box substitutions and XOR's together with an involution structure so that it can be efficiently implemented on various platforms.

1 Introduction

The block cipher Rijndael[1] has been selected as AES (Advanced Encryption Standard) by NIST in 2000[2] and also selected as a portfolio of NESSIE(New European Schemes for Signature, Integrity and Encryption) project with Camellia[3], MISTY1[4] and SHACAL-256[5] in 2003. The ciphers, Rijndael and Camellia have 128-bit block size and 128-, 192-, and 256-bit key lengths which are the interface specifications of AES, are considered as representative ciphers which satisfy such the interface. Rijndael[1] is an SPN cipher and uses operations such as multiplications in a finite field to have a diffusion effect of the states in 32-bits. Camellia[3] is a Feistel cipher and uses an 8×8 binary matrix to have a diffusion effect of the states in 64-bits.

In this paper, we propose a 128-bit block cipher ARIA which is an involution SPN which has been studied by AES developers. Khazad[6] and Anubis[7], which has been submitted to NESSIE project[8], are involution SPN ciphers and have involution substitution and diffusion layers by using operations such as multiplications in a finite field. Those variations are favorable in considering the efficiency, but some flaws[9] caused by the totally involution structure have been found. ARIA is an involution SPN block cipher which is not totally involutional. More precisely, the substitution layers are not involution. Therefore, the flaws given in [9] cannot be applied to ARIA. ARIA uses an involutional diffusion layer which is a 16×16 binary matrix. Also ARIA uses two kinds of S-boxes S_1, S_2 and two types of substitution layers $(LS, LS, LS, LS), (LS^{-1}, LS^{-1}, LS^{-1}, LS^{-1})$ where $LS = (S_1, S_2, S_1^{-1}, S_2^{-1})$. The substitution layers are arranged for ARIA to be an involution structure described as follows. In each odd round, the substitution layer is (LS, LS, LS, LS), in each even round, the substitution layer is $(LS^{-1}, LS^{-1}, LS^{-1}, LS^{-1})$. In addition to the involution SPN structure, we choose a diffusion layer to resist against powerful attacks which have been applied to reduced round Rijndael and Camellia, such as collision attacks[10], partial sum attacks[11], and truncated differential attacks. Since the size of the states mixed by the diffusion layers of Rijndael and Camellia are only a quarter or a half the size of a block, attacks which are listed the above are valid for some reduced rounds of those block ciphers. A diffusion layer which mixes all states is one of design principles and it is selected among 16×16 matrices. Note that a 16×16 matrix with entries $a_{ij} \in F$, where F is a finite field that is not $GF(2)$ is not a good choice because they are heavily involved with multiplications in a finite field F. Therefore 16×16 *binary* matrices are only candidates.

The maximum branch number of an invertible 16×16 binary matrix is *8* and the maximum branch number of an invertible 16×16 matrix with entries in a finite field is 17, which will be published later. We construct such an involution binary matrix of branch number *8*, and use some mathematical techniques to find a form in product of matrices, which is for increasing efficiency in 8-bit and 32-bit software implementations.

The cipher ARIA is an involution SPN cipher without having any weakness in S-boxes unlike Anubis and Khazad. The specifications of ARIA in detail are given in Section 2 and motivations of the design are given in Section 3. In Section 4, techniques of efficient implementation for 8-bit/32-bit processors are given. Security analysis against known attacks are given in Section 5.

2 Specifications

2.1 Notations

We use the following notations for describing ARIA.

- $S_i(x)$: The output of S-box $S_i (i = 1, 2)$ for an input x
- $A(x)$: The output of diffusion layer for an input x

- $\oplus$: A bitwise XOR operation
- $\|$: Concatenation of two operands
- $\ggg n$: Right circular rotation of operand by n bits
- $\lll n$: Left circular rotation of operand by n bits
- $\cdot$: Multiplication of two operands, matrix and vector, or two matrices

2.2 The round transformation

Each round of the cipher consists of the following three parts.

1. Round key addition: This is done by XORing the 128-bit round key.
2. Substitution layer: There shall be two types of substitution layers.
3. Diffusion layer: A simple linear map which is an involution.

Note that the diffusion layer of the last round is replaced by a round key addition.

Substitution layer We use 2 S-boxes S_1, S_2 and their inverses S_1^{-1}, S_2^{-1}. The each S-box is defined to be an affine transformation of the inversion function over GF(2^8).

$$S: \ \mathrm{GF}(2^8) \to \mathrm{GF}(2^8),$$
$$S_1\ x \mapsto A \cdot x^{-1} \oplus a,$$

where

$$A = \begin{pmatrix} 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1\\ 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1 \end{pmatrix} \quad \text{and} \quad a = \begin{pmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{pmatrix}.$$

and

$$S_2\ x \mapsto B \cdot x^{247} \oplus b,$$

where

$$B = \begin{pmatrix} 0&1&0&1&1&1&1&0\\ 0&0&1&1&1&1&0&1\\ 1&1&0&1&0&1&1&1\\ 1&0&0&1&1&1&0&1\\ 0&0&1&0&1&1&0&0\\ 1&0&0&0&0&0&0&1\\ 0&1&0&1&1&1&0&1\\ 1&1&0&1&0&0&1&1 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 0\\1\\0\\0\\0\\1\\1\\1 \end{pmatrix}.$$

The precalculated values of S_1, S_2 and S_1^{-1}, S_2^{-1} are given in Table 1 and Table 2. For example, $S_1(\texttt{0x00}) = \texttt{0x63}$, $S_1(\texttt{0x05}) = \texttt{0x6b}$, and $S_1(\texttt{0x72}) = \texttt{0x40}$.

ARIA has two types of S-box layers as shown in Figure 1.

The two types are used alternately and we use an even number of rounds so as to make the cipher involution. Type 1 is used in the odd rounds and type 2 is used in the even rounds.

Table 1. S-box S_1 and S_1^{-1}

S-box S_1

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-box S_1^{-1}

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

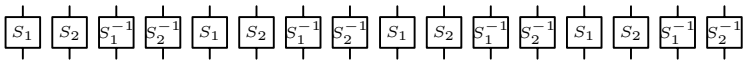
Table 2. S-box S_2 and S_2^{-1}

S-box S_2

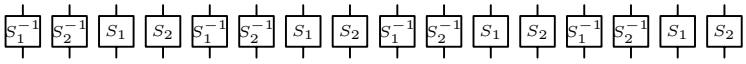
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	e2	4e	54	fc	94	c2	4a	cc	62	0d	6a	46	3c	4d	8b	d1
1	5e	fa	64	cb	b4	97	be	2b	bc	77	2e	03	d3	19	59	c1
2	1d	06	41	6b	55	f0	99	69	ea	9c	18	ae	63	df	e7	bb
3	00	73	66	fb	96	4c	85	e4	3a	09	45	aa	0f	ee	10	eb
4	2d	7f	f4	29	ac	cf	ad	91	8d	78	c8	95	f9	2f	ce	cd
5	08	7a	88	38	5c	83	2a	28	47	db	b8	c7	93	a4	12	53
6	ff	87	0e	31	36	21	58	48	01	8e	37	74	32	ca	e9	b1
7	b7	ab	0c	d7	c4	56	42	26	07	98	60	d9	b6	b9	11	40
8	ec	20	8c	bd	a0	c9	84	4	49	23	f1	4f	50	1f	13	dc
9	d8	c0	9e	57	e3	c3	7b	65	3b	02	8f	3e	e8	25	92	e5
a	15	dd	fd	17	a9	bf	d4	9a	7e	c5	39	67	fe	76	9d	43
b	a7	e1	d0	f5	68	f2	1b	34	70	05	a3	8a	d5	79	86	a8
c	30	c6	51	4b	1e	a6	27	f6	35	d2	6e	24	16	82	5f	da
d	e6	75	a2	ef	2c	b2	1c	9f	5d	6f	80	0a	72	44	9b	6c
e	90	b	5b	33	7d	5a	52	f3	61	a1	f7	b0	d6	3f	7c	6d
f	ed	14	e0	a5	3d	22	b3	f8	89	de	71	1a	af	ba	b5	81

S-box S_2^{-1}

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	30	68	99	1b	87	b9	21	78	50	39	db	e1	72	9	62	3c
1	3e	7e	5e	8e	f1	a0	cc	a3	2a	1d	fb	b6	d6	20	c4	8d
2	81	65	f5	89	cb	9d	77	c6	57	43	56	17	d4	40	1a	4d
3	c0	63	6c	e3	b7	c8	64	6a	53	aa	38	98	0c	f4	9b	ed
4	7f	22	76	af	dd	3a	0b	58	67	88	06	c3	35	0d	01	8b
5	8c	c2	e6	5f	02	24	75	93	66	1e	e5	e2	54	d8	10	ce
6	7a	e8	8	2c	12	97	32	ab	b4	27	0a	23	df	ef	ca	d9
7	b8	fa	dc	31	6b	d1	ad	19	49	bd	51	96	ee	e4	a8	41
8	da	ff	cd	55	86	36	be	61	52	f8	bb	0e	82	48	69	9a
9	e0	47	9e	5c	04	4b	34	15	79	26	a7	de	29	ae	92	d7
a	84	e9	d2	ba	5d	f3	c5	b0	bf	a4	3b	71	44	46	2b	fc
b	eb	6f	d5	f6	14	fe	7c	70	5a	7d	fd	2f	18	83	16	a5
c	91	1f	05	95	74	a9	c1	5b	4a	85	6d	13	07	4f	4e	45
d	b2	0f	c9	1c	a6	bc	ec	73	90	7b	cf	59	8f	a1	f9	2d
e	f2	b1	00	94	37	9f	d0	2e	9c	6e	28	3f	80	f0	3d	d3
f	25	8a	b5	e7	42	b3	c7	ea	f7	4c	11	33	03	a2	ac	60



(a) S-box layer type 1



(b) S-box layer type 2

Fig. 1. Two types of S-box layers

Diffusion layer The diffusion layer $A: \mathrm{GF}(2^8)^{16} \to \mathrm{GF}(2^8)^{16}$ is given by

$$(x_0, x_1, \ldots, x_{15}) \mapsto (y_0, y_1, \ldots, y_{15}),$$

where

$$\begin{aligned}
y_0 &= x_3 \oplus x_4 \oplus x_6 \oplus x_8 \oplus x_9 \oplus x_{13} \oplus x_{14}, & y_8 &= x_0 \oplus x_1 \oplus x_4 \oplus x_7 \oplus x_{10} \oplus x_{13} \oplus x_{15},\\
y_1 &= x_2 \oplus x_5 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{12} \oplus x_{15}, & y_9 &= x_0 \oplus x_1 \oplus x_5 \oplus x_6 \oplus x_{11} \oplus x_{12} \oplus x_{14},\\
y_2 &= x_1 \oplus x_4 \oplus x_6 \oplus x_{10} \oplus x_{11} \oplus x_{12} \oplus x_{15}, & y_{10} &= x_2 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_8 \oplus x_{13} \oplus x_{15},\\
y_3 &= x_0 \oplus x_5 \oplus x_7 \oplus x_{10} \oplus x_{11} \oplus x_{13} \oplus x_{14}, & y_{11} &= x_2 \oplus x_3 \oplus x_4 \oplus x_7 \oplus x_9 \oplus x_{12} \oplus x_{14},\\
y_4 &= x_0 \oplus x_2 \oplus x_5 \oplus x_8 \oplus x_{11} \oplus x_{14} \oplus x_{15}, & y_{12} &= x_1 \oplus x_2 \oplus x_6 \oplus x_7 \oplus x_9 \oplus x_{11} \oplus x_{12},\\
y_5 &= x_1 \oplus x_3 \oplus x_4 \oplus x_9 \oplus x_{10} \oplus x_{14} \oplus x_{15}, & y_{13} &= x_0 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{10} \oplus x_{13},\\
y_6 &= x_0 \oplus x_2 \oplus x_7 \oplus x_9 \oplus x_{10} \oplus x_{12} \oplus x_{13}, & y_{14} &= x_0 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_9 \oplus x_{11} \oplus x_{14},\\
y_7 &= x_1 \oplus x_3 \oplus x_6 \oplus x_8 \oplus x_{11} \oplus x_{12} \oplus x_{13}, & y_{15} &= x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_8 \oplus x_{10} \oplus x_{15}.
\end{aligned}$$

An equivalent expression would be given by a matrix multiplication as follow.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{15} \end{pmatrix} = \begin{pmatrix} 0&0&0&1&1&0&1&0&1&1&0&0&0&1&1&0 \\ 0&0&1&0&0&1&0&1&1&1&0&0&1&0&0&1 \\ 0&1&0&0&1&0&1&0&0&0&1&1&1&0&0&1 \\ 1&0&0&0&0&1&0&1&0&0&1&1&0&1&1&0 \\ 1&0&1&0&0&1&0&0&1&0&0&1&0&0&1&1 \\ 0&1&0&1&1&0&0&0&0&1&1&0&0&0&1&1 \\ 1&0&1&0&0&0&0&1&0&1&1&0&1&1&0&0 \\ 0&1&0&1&0&0&1&0&1&0&0&1&1&1&0&0 \\ 1&1&0&0&1&0&0&1&0&0&1&0&0&1&0&1 \\ 1&1&0&0&0&1&1&0&0&0&0&1&1&0&1&0 \\ 0&0&1&1&0&1&1&0&1&0&0&0&0&1&0&1 \\ 0&0&1&1&1&0&0&1&0&1&0&0&1&0&1&0 \\ 0&1&1&0&0&0&1&1&0&1&0&1&1&0&0&0 \\ 1&0&0&1&0&0&1&1&1&0&1&0&0&1&0&0 \\ 1&0&0&1&1&1&0&0&0&1&0&1&0&0&1&0 \\ 0&1&1&0&1&1&0&0&1&0&1&0&0&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{pmatrix} \tag{1}$$

2.3 Key schedule

The key schedule of ARIA consists of two parts, which are initialization and round key generation as follows.

Initialization In the initialization part, four 128-bit values W_0, W_1, W_2, W_3 are generated from the master key MK, by using a 3-round 256-bit Feistel cipher.

Note that MK can be of 128, 192, or 256 bit size. We first fill out the 128-bit value KL with bits from MK and use what is left of MK (if any) on the 128-bit value KR. The space remaining on KR (if any) is filled with zero as the follow.

$$KL||KR = MK||0 \cdots 0.$$

Then we set

$$\begin{aligned} W_0 &= KL, & W_2 &= F_e(W_1, CK_2) \oplus W_0, \\ W_1 &= F_o(W_0, CK_1) \oplus KR, & W_3 &= F_o(W_2, CK_3) \oplus W_1. \end{aligned}$$

Here, F_o and F_e are even and odd round functions, respectively, given in the previous section. The 128-bit keys CK_i of the round functions are fixed to be the rational part of π^{-1} and are given as follows.

$$\begin{aligned} CK_1 &= \texttt{0x517cc1b727220a94fe12abe8fa9a6ee0} \\ CK_2 &= \texttt{0x6db14acc9e21c820ff28b1d5ef5de2b0} \\ CK_3 &= \texttt{0xdb92371d2126e970324977504e8c90e0} \end{aligned}$$

This initialization process is given in Figure 2.

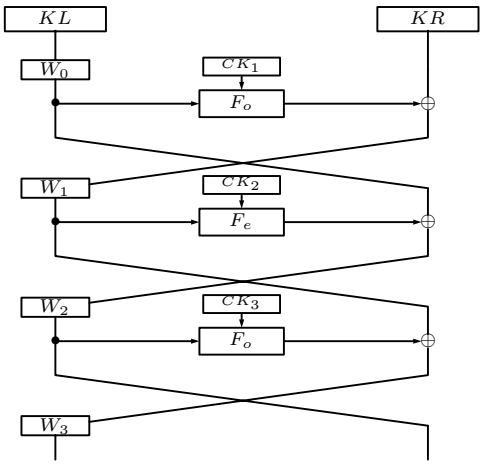


Fig. 2. Initialization part of key schedule

Round key generation In the generation part, combining the four values W_i to obtain an encryption round key ek_i and the decryption round key dk_i. Note that the number of rounds we use are 10, 12, or 14 which are depending on the size 128, 192, or 256 of the master key. Since there is one more key addition layer in the last round, 128-bit round keys are needed in the $11^{th}, 13^{th}$, or 15^{th} round,

depending on the size of master key. The value KR is also used in generating the round keys when the master key is of 256-bit size.

$$ek_1 = (W_0^{\gg 7}) \oplus (W_1^{\ll 11}), \qquad ek_2 = (W_1^{\ll 22}) \oplus (W_2),$$
$$ek_3 = (W_2^{\gg 17}) \oplus (W_3^{\ll 16}), \qquad ek_4 = (W_0^{\gg 14}) \oplus (W_3^{\ll 32}),$$
$$ek_5 = (W_0^{\gg 21}) \oplus (W_2^{\gg 34}), \qquad ek_6 = (W_1^{\ll 33}) \oplus (W_3^{\ll 48})$$
$$ek_7 = (W_1^{\ll 44}) \oplus (W_2^{\gg 51}), \qquad ek_8 = (W_0^{\gg 28}) \oplus (W_3^{\ll 64}),$$
$$ek_9 = (W_1^{\ll 55}) \oplus (W_3^{\ll 80}), \qquad ek_{10} = (W_0^{\gg 35}) \oplus (W_2^{\gg 68}),$$
$$ek_{11} = (W_0^{\gg 42}) \oplus (W_1^{\ll 66}), \qquad ek_{12} = (W_1^{\ll 77}) \oplus (W_2^{\gg 85}) \oplus (W_3^{\ll 96}),$$
$$ek_{13} = (W_0^{\gg 49}) \oplus (W_2^{\gg 102}), \qquad ek_{14} = (W_2^{\gg 119}) \oplus (W_3^{\ll 112}) \oplus (KR^{\ll 64}),$$
$$ek_{15} = (W_0^{\gg 56}) \oplus (W_1^{\ll 88}) \oplus (KR).$$

The decryption round keys are different from the encryption round keys and are derived from the encryption round keys. The ordering of round keys are reversed followed by the output of the diffusion layer A to all round keys except for the first and the last. The following equations represent how the decryption keys are computed by n which is given as the number of rounds.

$$dk_1 = ek_{n+1}, dk_2 = A(ek_n), dk_3 = A(ek_{n-1}), \cdots, dk_n = A(ek_2), dk_{n+1} = ek_1.$$

2.4 The cipher

The encryption and decryption processes of an n-round ARIA, where n is even, are given in Figure 3.

Note that the above two processes are identical except in the use of round keys.

2.5 Number of rounds

The number of rounds n can be a multiple of 2. We recommend the number of rounds of ARIA by 10-/12-/14-rounds for 128-/192-/256-bit keys, respectively.

3 Motivation for design choices

A cryptographic algorithm is usually designed so as to be implemented efficiently in both software and hardware. However, there is no set of rules one can follow to achieve this goal. We have designed ARIA taking into account many elements such as gate counts, memory requirements in smart card implementations, and software performance on multiple platforms.

The cipher consists only of four 8×8 substitution tables (S-boxes) and a linear transformation which can be efficiently implemented even in 8-bit low-end smart cards. The linear transformation, used as the diffusion layer, is an involution 16×16 binary matrix. It has the maximum branch number 8, and the number of ones has been minimized so as to reduce the total number of logical operations. We use basic logical operations in order to ease of hardware implementations.

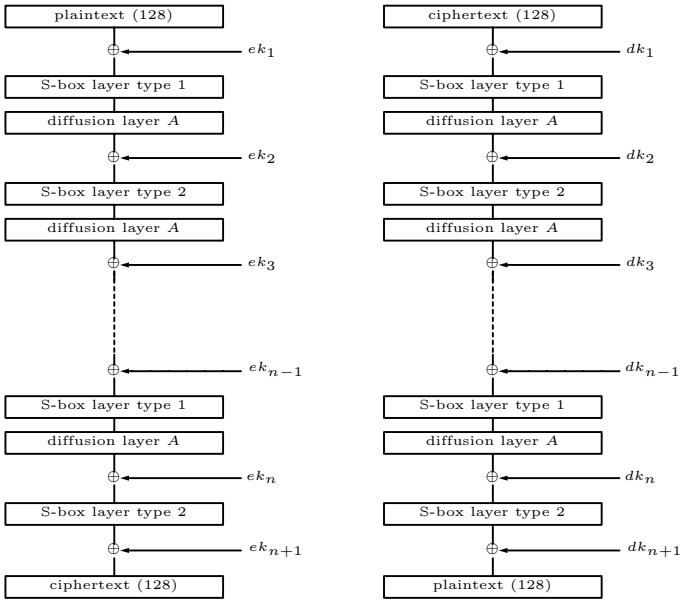


Fig. 3. Encryption and decryption processes

3.1 S-box

We design the S-boxes to meet the following criteria.

- The size of input and output is 8. i.e., it should be a function over GF(2^8).
- Degree of the Boolean polynomial describing each output bit is 7.
- It has the smallest of the maximum probability 2^{-6} for differential characteristic and linear approximation.

The S-box S on the inversion function over GF(2^8) is used to satisfy the above criteria. High degree of the Boolean polynomial representing each output bit of the S-boxes makes difficult to be applied higher order differential attacks on the cipher.

An affine transformation which is an 8×8 binary matrix followed by the inversion function is used for getting rid of fixed points and is chosen with the maximum branch number 5.

3.2 Diffusion layer

In designing a cipher, choosing a diffusion layer is important in both efficiency and security aspects. Since we want the cipher to be involutional and all states to be mixed by the diffusion layer, we search a matrix in 16×16 binary matrices. We choose a matrix which satisfies the following properties.

- It should be involutional.
- The branch number should be maximal.
- It should be suitable for implementations on 8-bit/32-bit processors.
- It should be suitable for hardware implementations.

The branch number $\beta(A)$ of a $n \times n$ diffusion layer A over a finite field F is defined by

$$\beta(A) = \min\{wt(x) + wt(Ax^T) | x \in F^n, x \neq 0\},$$

where $wt(x)$ is the Hamming weight of x.

We proved that the maximum branch number of invertible 16×16 binary matrices is 8 while the maximum branch number of invertible 16×16 matrices over finite field $GF(2^8)$ is 17. This will be published later. Since the matrix should be involutional, we searched the matrices of the form

$$M_1^{-1} \cdot M_2 \cdot M_1$$

where M_2 is an involution block diagonal matrix whose diagonal consists of four 4×4 binary matrices and M_1 is any 16×16 binary matrix which can be written as a 4×4 binary matrix on the vectors with four 32-bits block entries. The restriction are set for the efficiency on 32-bit processors.

For many matrices of the above form, we examined whether the branch number is 8 and the Hamming weights of each column is 7. More description of the matrix will be given in Section 4 and in [12].

3.3 Key expansion

The key schedule is designed by the following criteria.

- Input to the key schedule is of 128, 192, or 256 bit size.
- Only the basic logical operations are used.
- Resistance to related key attacks.
- It should be difficult to recover the master key from a small fraction of round keys.
- Delay time to the encryption or decryption process caused by key schedule should be minimized.

We use a 256-bit Feistel type cipher whose core function is the same as our round function to obtain four 128-bit values from master key. Then each round key is derived from two of the four values and the master key by rotating and XORing the 128-bit values. The size of each rotation was designed so as to deter one from obtaining one round key from a combination of other round keys.

In the key expansion part, the values W_0, W_1, W_2, W_3 and the rotation bits were chosen to satisfy the following conditions.

- In any two consecutive rounds, at least one of the 128-bit values used to generate the round keys is not common.

– For any two round keys that were generated from the same set of 128-bit values, one should not be able to recover a part of one round key from a part of the other round key.
– Calculating the first round key should cause as little as possible delay time to the encryption or decryption process.

4 Implementation aspects

ARIA is suitable for efficient implementations on various environments, especially in hardware. We present some techniques[12] which optimize implementations on 8-bit processors such as Smart Cards and on 32-bit processors such as PCs.

4.1 8-bit based implementations

On an 8-bit processor, except for S-box substitution, all operations are XOR's. The implementation of S-box substitution requires four tables of 256 bytes.

In a straight coding, 112 XOR's are required to implement one round except for S-box substitutions. We present a method to reduce the number of operations to 76 XOR's using four additional variables $T_1, \cdots, T_4$ as follows.

$$
\begin{array}{ll}
T_1 = x_4 \oplus x_5 \oplus x_{10} \oplus x_{15}, & T_2 = x_3 \oplus x_6 \oplus x_9 \oplus x_{16}, \\
y_1 = x_7 \oplus x_9 \oplus x_{14} \oplus T_1, & y_2 = x_8 \oplus x_{10} \oplus x_{13} \oplus T_2, \\
y_6 = x_2 \oplus x_{11} \oplus x_{16} \oplus T_1, & y_5 = x_1 \oplus x_{12} \oplus x_{15} \oplus T_2, \\
y_{12} = x_3 \oplus x_8 \oplus x_{13} \oplus T_1, & y_{11} = x_4 \oplus x_7 \oplus x_{14} \oplus T_2, \\
y_{15} = x_1 \oplus x_6 \oplus x_{12} \oplus T_1, & y_{16} = x_2 \oplus x_5 \oplus x_{11} \oplus T_2,
\end{array}
$$

$$
\begin{array}{ll}
T_3 = x_2 \oplus x_7 \oplus x_{12} \oplus x_{13}, & T_4 = x_1 \oplus x_8 \oplus x_{11} \oplus x_{14}, \\
y_3 = x_5 \oplus x_{11} \oplus x_{16} \oplus T_3, & y_4 = x_6 \oplus x_{12} \oplus x_{15} \oplus T_4, \\
y_8 = x_4 \oplus x_9 \oplus x_{14} \oplus T_3, & y_7 = x_3 \oplus x_{10} \oplus x_{13} \oplus T_4, \\
y_{10} = x_1 \oplus x_6 \oplus x_{15} \oplus T_3, & y_9 = x_2 \oplus x_5 \oplus x_{16} \oplus T_4, \\
y_{13} = x_3 \oplus x_8 \oplus x_{10} \oplus T_3, & y_{14} = x_4 \oplus x_{70} \oplus x_9 \oplus T_4.
\end{array}
$$

If it is implemented serially, only one extra variable is required in the method.

4.2 Software implementations on 32-bit processors

On 32-bit processors, Rijndael and Camellia can be implemented efficiently by combining S-box substitutions and the diffusion layer by 8×32 table lookups. This technique is suitable for a diffusion layer which is based on 32-bits. Since the diffusion layer of ARIA is based on 128-bits, on 32-bit processors, it looks less efficient than Rijndael and Camellia. However, we choose a matrix in 16×16 binary matrices which can be efficiently implemented on 32-bit processors.

Since M_1 is an involution($M_1^{-1} = M_1$ in Section 3.2) The diffusion layer A is chosen in the form of $M_1 \cdot M_2 \cdot M_1$ to have an involution structure where

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

for the 4×4 identity matrix I and following four 4×4 matrices

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

For simplifying the above notations, we use the following notations

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

If we let S be the S-box substitution, the one round except for key addition can be written as $A \cdot S = M_1 \cdot M_2 \cdot M_1 \cdot S$. It is easy to see that the form of $M_1 \cdot S$ is not implemented efficiently by 8×32 table lookups. So, we make the following modification for the representation of the diffusion layer as follows:

$$\begin{aligned} A = M_1 \cdot M_2 \cdot M_1 &= M_1 \cdot P \cdot M \cdot M_1 = M_1 \cdot P \cdot M \cdot M_1 \cdot M \cdot M \\ &= M_1 \cdot P \cdot M \cdot M \cdot M_1 \cdot M = M_1 \cdot P \cdot M_1 \cdot M. \end{aligned}$$

Since M is a block diagonal matrix, $M \cdot S$ can be implemented by 8×32 table lookups. It is clear that M_1 is implemented by simple 32-bit word operations. The operation P is a byte-oriented operation that is done within each 32-bit word. Hence this gives a way to implement ARIA on a 32-bit based machine. On 32-bit processors, the encryption speed of ARIA is at least 70% of that of Rijndael. Notice that there are only three or four ciphers submitted in AES and NESSIE project whose encryption speeds are in the above range. Therefore we can see that ARIA is also efficient on 32-bit processors.

4.3 Remark on the hardware implementations

The hardware length is mainly determined by the number of S-box layers unless diffusion layers are too heavy. Since the number of S-box layers are same as that of Rijndael and smaller than that of Camellia, throughput is almost same as that of Rijndael and faster than that of Camellia. Since ARIA is involutional, it requires only one procedure to be implemented for the encryption and the decryption unlike Rijndael. Therefore the area to be implemented in hardware for ARIA is smaller than that of Rijndael, i.e., in considering the efficiency, ARIA is better than Rijndael.

5 Strength against known attacks

5.1 Differential and linear cryptanalysis

The maximum differential probability p and the maximum linear probability q of the S-box S for ARIA is given by

$$p = q = 2^{-6}.$$

The branch number β_d in respect to the differential cryptanalysis and the branch number β_l in respect to the linear cryptanalysis for the diffusion layer A of ARIA are given by

$$\beta_d = \beta_l = 8.$$

The minimum number of active S-boxes with respect to differential and linear cryptanalysis in r-rounds is

$$8 \cdot \lfloor r/2 \rfloor + 2 \cdot (r/2 - \lfloor r/2 \rfloor).$$

The upper bound on differential characteristic probabilities and linear approximation probabilities for 6-rounds is $(2^{-6})^{24} = 2^{-144}$. Therefore, there are no effective differential characteristics and linear approximations for ARIA in 6 or more rounds.

The upper bound on differential characteristic probabilities and linear approximation probabilities for 5-rounds is $(2^{-6})^{17} = 2^{-102}$ which is greater than 2^{-128}. Even if an attacker uses the 5-round differential characteristic or linear approximation on ARIA we expect 8 or more rounds to provide sufficient resistance against differential and linear cryptanalysis.

5.2 Truncated differential cryptanalysis

By computational experiments, we searched effective truncated differential characteristics which distinguish the reduced-rounds of ARIA from a random permutation. We found many effective truncated differential characteristics on 5 round variant of ARIA which can be used to attack 7 rounds. However, there are no effective truncated differential characteristics on 6 or more rounds. Thus, we think that ARIA has sufficient security against truncated differential cryptanalysis.

5.3 Impossible differential cryptanalysis

We have confirmed that there are no bytes whose difference is always zero(or nonzero) after 2-rounds of encryption. Also, We have checked that there are no bytes whose difference is always zero or nonzero after 2-rounds of decryption. Therefore, we expect that there are no impossible differentials on 4 or more rounds.

5.4 Square attack(Integral cryptanalysis)

Applying the square attack on ARIA, we considered a collection of plaintexts in which each byte is either active or passive.

In case of ARIA, for any collection of plaintexts, determining whether any given byte position is balanced or not after 3-rounds of encryption is not possible. Therefore, one can construct only 2-round distinguishers.

5.5 Higher order differential cryptanalysis

The S-boxes S and S^{-1} of ARIA have algebraic degree 7. Each output bit of the S-box can be regarded as a Boolean function with 8 input variables. After three rounds the algebraic degree of any intermediate bit becomes 7^3. Thus, the number of plaintexts needed for higher order differential attack using a 3 round distinguishers is greater than 2^{128}. So only up to 2 round distinguisher may be found for application of higher order differential attack.

5.6 Interpolation attack

In the interpolation attack, using plaintext and ciphertext pairs, the attacker constructs polynomials describing relations between the input and output to the cipher. If the constructed polynomials have small degree, only a small number of plaintexts and their corresponding ciphertexts are necessary to solve for the coefficients of the polynomial. This is done through the use of Lagrange interpolation formula[14].

We expect that the complicated expression of the S-boxes used in ARIA, together with the effect of the diffusion layer will prohibit the interpolation attack on more than just a few rounds.

5.7 Weakness in the key schedule

Since the round keys in ARIA are always applied using the XOR operation, one cannot find a weak key class as in IDEA. Also, we expect that there are no equivalent keys. Thus there is no restriction on key selection. The key schedule of ARIA has sufficient number of nonlinear components so as to avoid the related key attack or any other attacks rising from the weakness of the key schedule. In particular, it is impossible to calculate the encryption key from partial information of round keys.

6 Conclusion

We have proposed a new 128-bit block cipher ARIA based on SPN structure. ARIA is a simple and elegant algorithm with the following properties:

- uses only basic operations such as XOR and S-box substitution.

- uses a 16×16 binary matrix with branch number 8(maximal) in diffusion layer.
- adopts an involutional structure for efficient implementations on multiple environments.
- does not adopt a totally involutional structure to avoid flaws found in Khazad and Anubis.

We could not find any significant weakness and have not inserted any hidden weakness. We think that ARIA is suitable for most platforms and can be widely used.

References

1. Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer, 2001.
2. NIST, NIST announces that Rijndael has been selected as the proposed AES. October 2, 2000. Available at http://csrc.nist.gov/CryptoToolkit/aes/
3. Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis, LNCS 2012 , pages 39–56. Springer, 2000.
4. M. Matsui, *Block Encryption Algorithm MISTY*, *Fast Software Encryption*, 4th InternationalWorkshop Proceeding, LNCS 1267, Springer-Verlag, pp.54-68, 1997.
5. H.Handschuh, D.Naccache, SHACAL, In proceedings of the First Open NESSIE Workshop, November 2000.
6. P. S. L. M. Barreto and V. Rijmen, *T*he Khazad legacy-level block cipher. Primitive submitted to NESSIE, Sept. 2000.
7. P. S. L. M. Barreto and V. Rijmen, *T*he Anubis block cipher. Primitive sub- mitted to NESSIE, Sept. 2000.
8. NESSIE Project, New European Schemes for Signatures, Integrity and Encryption, Homepage-avaiable at http://cryptonessie.org.
9. A. Biryukov, *Analysis of Involutional Ciphers: Khazad and Anubis*, In Fast Software Encryption, Proceedings of FSE 2003.
10. H. Gilbert and M. Minier, *A collision attack on seven rounds of Rijndael*, Proceeding of the third AES conference, pp230–241, NIST, 2000.
11. N. Ferguson, J. Kesley, S. Lucks, B. Schneier, M. Stay, D. Wagner and F. Whiting, *Improved Cryptanalysis of Rijndael*, In Fast Software Encryption, Proceedings FSE 2000, LNCS #1978, pp. 213–230, Springer-Verlag, 2000.
12. BonWook Koo, HwanSeok Jang, JungHwan Song. Constructing and Cryptanalysis of a 16x16 Binary Matrix as a Diffusion Layer. , editors, *WISA2003*, *Lecture Notes in Computer Science*, Springer, 2003.
13. David Wagner. The boomerang attack. In Lars R. Knudsen, editor, *Preproceedings of Fast Software Encryption Workshop 1999*, pages 155–169, 1999.
14. Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In Eli Biham, editor, *Preproceedings of Fast Software Encryption Workshop 1997*, pages 28–40, 1997.