블록 암호 알고리즘 ARIA

ARIA 개발팀*

Block Encryption Algorithm ARIA

ARIA Development Team

요 약

미국, 유럽, 일본 등에서의 공모 사업을 통하여 많은 블록 암호들이 개발되었으나, 국내에서 개발된 블록 암호들은 크게 주목을 끌지 못했다. 블록 암호 설계에 있어서 기존 암호와의 차별성, 안전성, 각종 플랫폼에서의 효율성이 중시되는데 이러한 조건을 만족하는 것은 쉽지 않기 때문이다.

본 논문에서는 16×16 이진 행렬을 이용한 확산 계층을 사용하여 기존 알고리즘들(AES, Camellia 등)과 차별성을 두고, AES에 적용된 공격들에 대한 내성을 갖는 블록 암호 알고리즘 ARIA를 제안한다.

ARIA는 간단한 연산들을 사용하여 8-비트 및 하드웨어 환경에서 AES 및 Camellia와 동등하거나 그 이상의 수행속도를 보인다. 이러한 특성을 이용하면, 스마트 카드 등과 같은 저전력, 초경량 환경과 ASIC 구현을 통한 초고속 환경에 활용될 수 있을 것이다.

1. 서론

정보 통신 서비스의 다변화 및 전자 정부 구현 등으로 전산망을 통한 국가기관과 민간(G2C)간 소통 자료가 급증하고 있으며, 이에 따라 국가 기관과 민간을 연결하는 암호 제품 및 이러한 용도에 적합한 안전성과 효율성을 제공하는 암호 알고리즘에 대한 수요가 증가하고 있다.

본 논문은 블록 암호 알고리즘 ARIA를 소개한다. ARIA는 SPN(Substitution-Permutation Network) 구조의 128 비트 블록 암호이며, 128 비트, 192 비트, 256 비트의 3 종류의 키 사용을 제공한다. ARIA의 입,출력 크기와 사용 가능한 키 크기는 미국 표준 블록 암호인 AES(Advanced Encryption Standard) [9]의 입,출력 크기 및 사용 가능한 키 크기와 동일하다.

블록 암호는 대표적인 공격 방법인 차분 공격(differential cryptanalysis) [5]과 선형 공격(linear cryptanalysis) [17]에 대하여 안전하여야 한다. 또한, 부정 차분 공격(truncated differential cryptanalysis) [15], 불능 차분 공격(impossible differential cryptanalysis) [3], 고계 차분 공격(higher order differential cryptanalysis) [15, 16], square 공격 [7] 등의 다양한 공격 방법에 대하여 안전하여야 한다. ARIA는 이러한 공격 방법들에 대하여 6 라운드 이상에서 2^{-128} 보다 큰 확률을 가지는 어떠한 특성도 존재하지 않으며, 이는 ARIA가 여러 공격 방법들에 안전함을 의미한다.

ARIA는 AES, Camellia [1]와는 달리 128 비트 전체 블록을 처리하는 확산 단계(diffusion layer)를 사용한다. AES의 확산 단계인 MixColumn 연산은 32 비트 블록(전체 블록의 1/4)을 처리하며, Camellia의 확산 단계는 64 비트 블록(전체 블록의 1/2)을 처리한다. ARIA의 이러한 특성은 collision 공격 [11], partial sum 공격 [10], 부정 차분 공격에 대하여 AES와 Camellia 보다 강한 안전성을 제공한다.

ARIA의 암호부는 8×8 S-box와 EXOR 연산만으로 구성되어 있으며, AES 블록 암호에 사용된 유한체 연산을 사용하지 않는다. 따라서, ARIA는 8 비트 플랫폼의 소프트웨어 환경과 ASIC 등의 하드웨어 환경에서 고속으로 동작하는데, AES 및 Camellia와 동급 또는 그 이상의 속도를 제공한다.

본 논문의 구성은 다음과 같다. 2장은 구조 및 동작 방법을, 3장에서는 설계 사상을 소개한다. 4장에서는 ARIA의 각종 공격 방법에 대한 안전성을 5장에서는 여러 구현 환경에서의 구현 효율성을 기술한다.

*권대성[1],김재성[2],박상우[1],성수학[3],손예권[2],송정환[4],염용진[1],윤이중[1],이상진[5],이제원[2],지성택[1],홍진[1]([1]NSRI,[2]국제과학문화연구소,[3]배재대학교,[4]한양대학교,[5]고려대학교)

2. 알고리즘 구조

가. 개요 및 기호

블록 암호 알고리즘 ARIA의 특징은 다음과 같다.

- SPN(Substitution-Permutation Network) 구조
- 128-비트 입출력
- 128/192/256-비트 키
- 10/12/14 라운드
- Involution 구조

본 고에서는 다음과 같은 기호를 사용한다.

- $S(x)$: 입력 x에 대한 S-box 출력
- $A(x)$: 입력 x에 대한 Diffusion Layer의 출력
- $\oplus$: 비트별 EXOR 연산
- $\|$: 연접
- $\ggg n$: n비트 오른쪽 rotation
- $\cdot$: 행렬과 행렬 또는 벡터의 곱

나. 라운드 함수

각 라운드 함수는 다음과 같은 세부분으로 구성되어 있다.

1. 라운드 키 덧셈 : 128-비트 라운드 키를 입력과 EXOR 한다.
2. 바이트 치환 : 두 종류의 8-비트 치환 테이블을 이용하여 16바이트를 각각 치환한다.
3. 확산 계층 : Involution 구조의 16×16 이진 행렬을 이용하여 바이트들을 섞는다.

마지막 라운드에서는 확산 계층이 생략되며 키 덧셈이 한번 추가된다.

1) 바이트 치환

ARIA는 하나의 s-box S와 그의 역치환 S^{-1}를 사용한다. S-box는 유한체 $\mathrm{GF}(2^8)$ 상의 x^{-1}와 아핀변환으로 정의된다.

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

$$B = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

<표 1> s-box S

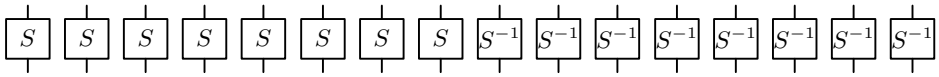
39	4d	8c	e7	e3	81	52	fb	db	7d	ea	d8	88	96	d4	53
4f	e0	90	71	5b	14	cd	5c	6a	76	66	07	cc	9b	80	a8
06	4c	59	b7	65	fa	91	5d	08	fd	ab	f4	cb	29	03	ee
1c	c1	9a	79	1d	0c	ae	48	c3	1a	6f	0a	e2	5a	f1	2f
a6	3c	84	58	82	25	f5	e8	9f	b5	dc	df	6d	28	0b	f6
2d	6e	d7	64	73	0e	57	11	c8	e4	be	f9	27	b2	d2	da
23	8e	ca	ed	67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

<표 2> s-box S^{-1}

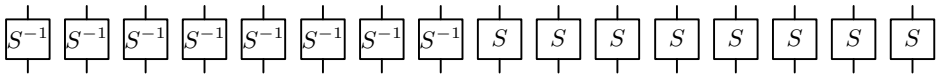
7a	f3	91	2e	f2	c1	20	1b	28	ac	3b	4e	35	6e	55	d3
69	57	81	c5	15	c8	f1	98	75	66	39	a8	30	34	df	cd
72	ff	d2	60	9f	45	bb	5c	4d	2d	ae	68	f9	50	aa	3f
bd	da	d5	b7	d0	b8	e3	a0	6d	00	c3	73	41	6b	e4	ef
70	65	e7	83	94	79	f8	be	37	ee	b0	b3	21	01	a4	10
c6	93	06	0f	d8	7f	f7	56	43	22	3d	14	17	27	b6	8c
e1	f6	e2	dd	53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

S와 S^{-1}는 <표 1>, <표 2>에 주어져 있으며 다음과 같이 읽는다. 예를 들면, $S(\texttt{0x00}) = \texttt{0x39}$, $S(\texttt{0x05}) = \texttt{0x81}$, $S(\texttt{0x72}) = \texttt{0x20}$.

ARIA는 두 종의 바이트 치환 계층을 가지고 있으며 (그림 1)과 같이 주어진다. 두 종의 치환



(a) S-box layer type 1



(b) S-box layer type 2

(그림 1) 2종의 바이트 치환 계층

계층은 교대로 사용된다. 따라서, 전체 구조가 involution이 되기 위해서는 전체 라운드 수가 짝수가 되어야 한다. Type 1은 홀수번째 라운드에 Type 2는 짝수번째 라운드에 사용된다.

2) 확산 계층(Diffusion layer)

Diffusion layer A는 다음과 같이 주어진다.

$$A: \ \mathrm{GF}(2^8)^{16} \to \mathrm{GF}(2^8)^{16},$$

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

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

A는 다음과 같은 16×16 이진 행렬로 표현할 수도 있다.

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

A가 involution임은 쉽게 보일 수 있다.

다. 키 스케줄

ARIA의 키 스케줄은 두 부분으로 구성되어 있다. 먼저, 키 초기화 과정을 거친 후 키 생성 과정에서 각 라운드의 키를 얻는 구조이다.

1) 초기화 과정

초기화 과정에서는 3-라운드 256-비트 Feistel 알고리즘을 이용하여, 마스터 키 MK로 부터 네개의 128-비트 값 W_0, W_1, W_2, W_3를 생성한다.

MK는 128-, 192-, 또는 256-비트인데, 먼저 MK의 128-비트를 KL이라 하고 남은 부분에 필요하면 0을 추가하여 128-비트 KR를 만든다.

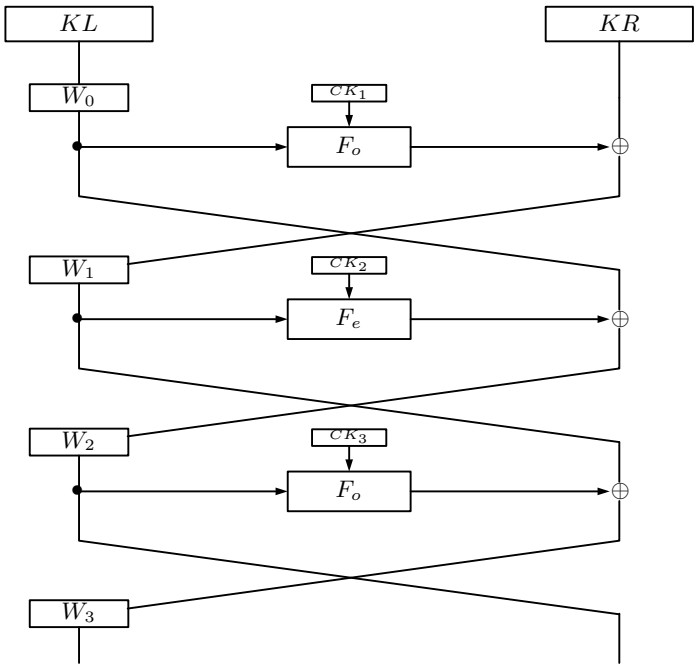
$$KL||KR = MK||0 \cdots 0.$$

네 값 W_0, W_1, W_2, W_3는 다음과 같이 생성한다.

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

여기에서, F_o, F_e는 각각 알고리즘의 홀수번째, 짝수번째 라운드 함수로, F_o는 type 1의 바이트 치환 계층을, F_e는 type 2의 바이트 치환 계층을 사용한다. 위 라운드 함수에 필요한 128-비트 키 CK_1, CK_2, CK_3는 π^{-1}의 유리수 부분을 128비트 씩 차례로 취한다.

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



(그림 2) 키 스케줄 초기화 과정

2) 라운드 키 생성

라운드 키 생성 과정에서는 네 값 W_0, W_1, W_2, W_3를 조합하여 암호화 라운드 키 ek_i와 복호화 라운드 키 dk_i를 얻는다. 128-, 192-, 256-비트 키에 대하여 ARIA의 라운드 수로 10, 12, 14라운드를 제안하므로, 각각 11, 13, 15개의 라운드 키를 생성한다. 256-비트 키의 경우에는 네 값 외에 KR를 추가로 사용하여 이전 라운드 키들과 연관 관계가 적도록 하였다..

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

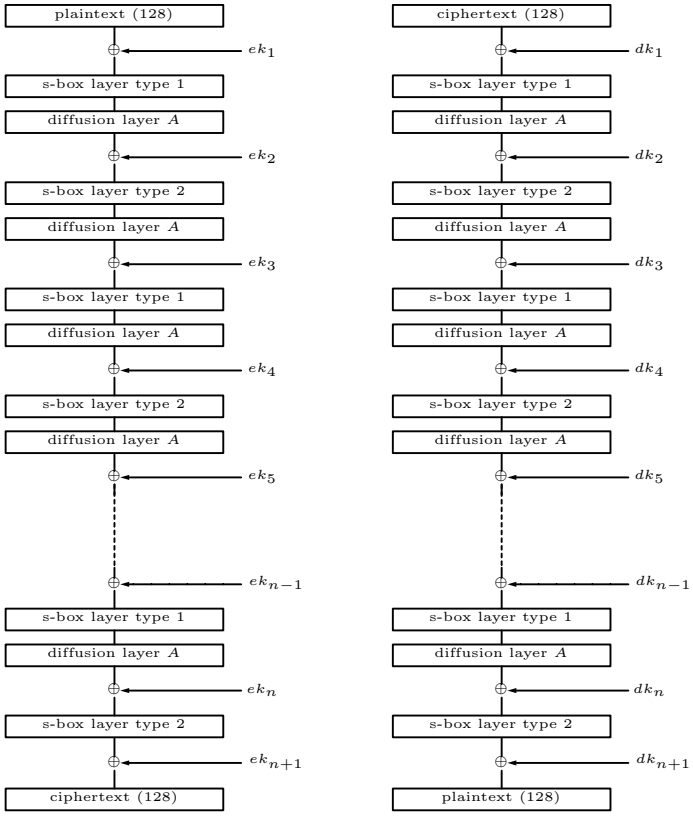
복호화 라운드키는 dk_i는 암호화 라운드 키 ek_i의 순서를 뒤집고 첫 라운드와 마지막 라운드를

제외한 모든 라운드의 키에 Diffusion Layer A를 적용하여 얻을 수 있다.

$$\begin{aligned} dk_1 &= ek_{n+1}, \\ dk_2 &= A(ek_n), \\ dk_3 &= A(ek_{n-1}), \\ &\vdots \\ dk_n &= A(ek_2), \\ dk_{n+1} &= ek_1. \end{aligned}$$

라. 전체 구조

n-라운드 ARIA의 암호화 과정과 복호화 과정은 (그림 3)과 같이 주어진다. 두 과정은 라운드 키



(그림 3) 암호화 과정과 복호화 과정

사용을 제외하고는 일치한다.

3. Design rationale

블록 암호 알고리즘 ARIA는 ASIC 등의 하드웨어 구현에서의 효율성을 위하여 게이트 수를, 스마트 카드 등의 소프트웨어 구현에 적합하도록 메모리 요구량을 고려하고 또한 각종 플랫폼에서의 속도를 고려하여 설계되었다.

ARIA는 8×8-비트 치환 테이블(s-boxes)과 하드웨어 구현이 효율적인 선형변환으로만 이루어져 있다. 선형변환은 16×16 involution 이진 행렬 중에서 최대 확산 효과를 가지며 최소의 연

산량을 가지는 것을 선택하였다. 이러한 연산들은 확산 효과를 높이면서도 소프트웨어 및 하드웨어 구현에 있어서 효율적인 것들이다.

가. S-box

S-box들이 다음을 만족하도록 설계하였다.

- 입출력 크기 : 8비트
- 대수적 차수 : 7.
- 최대 차분/선형 확률 : 2^{-6}
- 고정점/반고정점이 없음

ARIA의 치환 부분은 하나의 치환 함수를 이용한 치환 또는 역치환으로 구성되어 있으며 치환 함수는 AES, Camellia에서와 같이 GF(2^8)상에서의 x^{-1}와 아핀 변환의 합성으로 이루어져 있다. 아핀 변환은 고정점/반고정점이 없도록 선택되었으며 또한 비트 단위 확산 효과를 고려하였다.

나. 확산 계층(The Diffusion Layer)

16×16 이진 행렬 중에서 다음의 성질을 만족하는 것을 선택하였다.

성질 1 Branch number = 8.

성질 2 다양한 플랫폼에서 효율적으로 구현 가능

성질 1, 성질 2을 만족하도록 다음과 같은 4×4 이진 행렬들을 이용하였다.

$$P_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, P_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, P_3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, P_4 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$I_4 = P_0$를 4×4 단위 행렬이라고 할 때, 4×4 단위 행렬들을 entry로 하는 다음과 같은 4×4 행렬(16×16)들을 선택한다.

$$M_1 = \begin{pmatrix} 0 & I_4 & I_4 & I_4 \\ I_4 & 0 & I_4 & I_4 \\ I_4 & I_4 & 0 & I_4 \\ I_4 & I_4 & I_4 & 0 \end{pmatrix}, M_2 = \begin{pmatrix} P_0 & 0 & 0 & 0 \\ 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 \\ 0 & 0 & 0 & P_3 \end{pmatrix}, M_3 = \begin{pmatrix} P_4 & 0 & 0 & 0 \\ 0 & P_4 & 0 & 0 \\ 0 & 0 & P_4 & 0 \\ 0 & 0 & 0 & P_4 \end{pmatrix}$$

위 행렬들은 모두 involution이고 $M_1 \cdot M_3 = M_3 \cdot M_1,\ M_2 \cdot M_3 = M_3 \cdot M_2$의 관계를 만족한다. Diffusion Layer A를 $M_1 \cdot M_2 \cdot M_3 \cdot M_1$으로 선택하면 다음과 같은 성질이 있다.

- $A^{-1} = A$
- $A = M_1 \cdot M_2 \cdot M_1 \cdot M_3 = M' \cdot M_3$, 32비트 소프트웨어 구현에 효율적
- Branch number = 8.

다. 키 스케줄

키 스줄은 다음의 성질을 만족하도록 설계되었다.

- 키 스케줄 입력은 128-, 192-, 또는 256-비트가 되도록 한다.
- 되도록 간단한 연산을 사용한다.
- 연관키 공격에 안전해야 한다.
- 라운드 키로부터 마스터 키를 알기 어려워야 한다.
- 적은 지연시간으로 암호화/복호화가 진행되어야 한다.

128-, 192-, 256-비트 마스터 키를 입력으로 하고, 충분한 전파 특성을 얻기 위하여 알고리즘의 라운드 함수와 같은 라운드 함수를 이용한 3라운드 Feistel 구조를 초기화 과정에 사용하였다. 마스터 키와 각 라운드의 출력에서 128비트를 선택하여 4개의 값으로 놓고, 각 라운드 키는 이 중 두개를 선택하여 각각 rotation 후 EXOR을 취하여 생성한다. 마스터 키가 256-비트인 경우에는 마스터 키의 나머지 128-비트를 추가적으로 EXOR하는 방식을 사용하였다.

키 스케줄에서 값(W_0, W_1, W_2, W_3)의 선택과 rotation 양의 선택은 다음이 만족되도록 하였다.

- 연속된 두 라운드에서 사용된 값은 적어도 하나는 달라야 한다.
- 같은 두 값을 사용하여 생성한 두 라운드 키의 일부분으로 두 값의 일부를 예측할 수 없어야 한다.
- 암호화/복호화 시 첫 라운드 생성에 소요되는 시간이 적어야 한다.

라. 라운드 수

블록 암호 알고리즘 ARIA는 5라운드 이하에서 2^{-128} 이상의 확률을 가진 차분/선형 특성이 존재하며 전파 특성을 고려하면 마스터 키가 128-비트의 경우 8라운드, 192-비트의 경우 10라운드, 256-비트의 경우 12라운드 이상에서 전수조사보다 효율적인 차분/선형 공격이 존재하지 않을 것으로 예상된다. 현재까지 알려진 공격 방법에 대해서도 위 라운드에서 전수조사보다 효율적인 것을 발견하지 못했다. 따라서, ARIA의 라운드 수는 각각 2라운드의 마진을 추가하여 다음과 같이 결정한다.

<표 3> 라운드 수

키 크기	128-비트	192-비트	256-비트
DC/LC에 안전한 라운드 수	8	10	12
기타 공격에 안전한 라운드 수	8	10	12
ARIA라운드 수	10	12	14

4. 안전성 분석

가. 차분/선형 공격

Biham and Shamir [5]에 의하여 시작된 차분공격과 Matsui [17]의 선형공격은 블록암호에 대한 가장 잘 알려진 공격법이다. 차분, 선형 공격에 대한 안전성은 각각 차분특성 확률과 선형근사 확률에 의존한다. 이 확률의 상한은 연속된 라운드에 대한 활성 S-box의 개수로부터 계산될 수 있다[8].

확률의 상한은 S-box의 특성과 선형 확산함수의 branch number에 의하여 결정할 수 있다. 우선, ARIA의 S-box에 대한 최대 차분확률 p와 선형근사 확률 q는 $p = q = 2^{-6}$이다. 또한 차분특성 관점의 branch number β_d와 선형근사 관점의 branch number β_l은 $\beta_d = \beta_l = 8$을 만족한다. 이로부터 r 라운드에 대한 활성 S-box의 최소 개수를 구하면 다음과 같다.

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

6라운드에 대한 확률은 $(2^{-6})^{24} = 2^{-144}$이므로 연속된 6라운드에 대한 차분, 선형 특성은 발견되지 않는다. 5라운드의 경우 확률은 $(2^{-6})^{17} = 2^{-102}$이 되어 2^{-128}보다 큰 확률을 얻으며 이 특성으로부터 128비트 키의 경우 8라운드면 안전할 것으로 판단된다.

나. 부정차분 공격

바이트 연산 위주로 설계된 블록암호의 경우 차분, 선형공격과 함께 부정차분 공격도 매우 중요하게 다루어질 필요가 있다. 라운드 함수의 부정차분 특성을 조사하여 부정차분 행렬을 구성하고 설계자의 관점에서 width-first search기법을 적용하여 다중 라운드에 대한 부정차분 확률의 상한을 계산하였다. 이 방법으로 5라운드 이후에는 유효한 부정차분 특성이 없음을 보였고 따라서 8라운드면 부정차분 공격에 대하여 충분한 내성을 가질 것으로 판단된다.

한편, 공격자의 관점에서 분석한 결과로 4라운드까지는 랜덤함수와 구별되는 부정차분 특성이 존재함을 확인하였다. 여기서 $HW(i)$는 헤밍무게가 i임을 나타낸다.

- 확률 2^{-96}: $HW(1) \rightarrow HW(7) \rightarrow HW(1) \rightarrow HW(7) \rightarrow HW(1)$
  - (1000 0000 0000 0000) $\rightarrow$ (0001 1010 1100 0110): $p_f = 1$
  - (0001 1010 1100 0110) $\rightarrow$ (1000 0000 0000 0000): $p_f = 2^{-48}$
- 확률 2^{-96}: $HW(2) \rightarrow HW(6) \rightarrow HW(2) \rightarrow HW(6) \rightarrow HW(2)$
  - (1000 0100 0000 0000) $\rightarrow$ (0100 0010 1010 0101): $p_f = 2^{-8}$
  - (0100 0010 1010 0101) $\rightarrow$ (1000 0100 0000 0000): $p_f = 2^{-40}$

공격자 관점에서는 4라운드 특성이 발견되므로 설계자 관점에서 구한 확률은 충분히 엄밀한 상한이다.

다. 불능차분 공격

발생할 수 없는 차분특성을 이용하는 불능차분 공격[4, 3]은 바이트 단위로 분석한다는 점에서 부정차분과 유사한 공격으로 볼 수 있다. 불능차분 특성으로부터 잘못된 키를 지워나가는 방식으로 올바른 키를 찾는 공격기법이다. 임의의 평문 패턴에 대하여 2라운드 이후에는 차분값이 항상 0이 되는 바이트가 존재하지 않는다. 따라서 3라운드 이상의 불능차분 특성을 구성할 수 없으며 ARIA는 불능차분 공격에 대하여 충분한 안전성을 갖는다.

라. Square 공격

블록암호 SQUARE에 처음으로 적용된 Square 공격[7]은 바이트 연산의 블록암호 분석에 유용하게 사용되어 왔다. Square 공격은 바이트 단위 혹은 워드 단위의 선택 평문 패턴이 전파되는 특성을 이용한 공격이다. 입력 평문 집합에서 모든 값이 한번씩 나타나는 바이트를 active(A)라 하며 고정된 값이 사용된 바이트를 passive 혹은 constant(c)라 한다. 여러 라운드를 거친 값을 모두 EXOR한 경우 0이 되면 balanced 특성(B)을 갖는다고 하며 이를 이용하여 랜덤함수와 구별한다.

입력 평문을 *Accc cccc cccc cccc*의 패턴으로 구성하면 2라운드 출력의 각 바이트는 balance이다.

$$Accc\ cccc\ cccc\ cccc \longrightarrow cccA\ AcAc\ AAcc\ cAAc \longrightarrow BBBB\ BBBB\ BBBB\ BBBB$$

이러한 특성은 3라운드 S-box 통과 후 더이상 유지되지 않기 때문에 Square 공격에 사용할 수 있는 특성은 2라운드 특성까지이다. 따라서 5라운드 이상에 대한 Square 공격은 불가능하다고 판단된다.

마. 부메랑 공격

부메랑 공격[20]은 하나의 긴 라운드 특성 대신 두개의 확률 높은 짧은 라운드 특성을 활용하는 공격법이다. 확률 p, q인 두개의 차분 특성을 연결하여 구성한 부메랑 공격이 전수조사보다 효과적일 조건은 다음 식으로 주어진다.

$$(pq)^2 \geq 2^{-128}.$$

<표 4>로부터 확률 2^{-108}의 3라운드 부메랑 distinguisher를 만들 수 있으며 4라운드 이상에서는 2^{-128}보다 큰 확률을 갖는 부메랑 distinguisher를 만들 수 없음을 알 수 있다. 따라서 6라운드 이상에서는 부메랑 공격에 대하여 안전하다고 판단된다.

<표 4> 라운드별 차분확률의 상한

라운드 수	1	2	3	4	5
활성 S-box의 최소 개수	1	8	9	16	16
확률의 상한	2^{-6}	2^{-48}	2^{-54}	2^{-96}	2^{-102}

바. 고계차분 공격

고계차분 공격은 Knudsen[15]에 의하여 제안된 차분공격의 확장이다. 부울함수 f의 차분은

$$\Delta_a f(x) = f(x \oplus a) \oplus f(x)$$

으로 주어지며 이를 확장하여 i차 차분을 다음과 같이 정의한다.

$$\Delta^i_{a_1,\ldots,a_n} f(x) = \Delta_{a_i}(\Delta^{(i-1)}_{a_1,\ldots,a_{i-1}} f(x)).$$

차수가 d인 다항식으로 표현되는 부울함수의 경우 d차 차분은 상수이고, $d+1$차 차분은 0임을 이용하여 랜덤함수와 구별되는 distinguisher를 만들 수 있다. 고계차분 공격은 이 distinguisher의 전후에 몇 라운드를 더하여 키를 찾는 방법이다. 고계차분 공격에 안전하기 위해서는 대수적 차수가 큰 논리를 사용하여야 한다. ARIA의 경우 S-box의 대수적 차수는 7로 가능한 최대의 값이다. 3라운드를 거치면 대수적 차수가 7^3이 되어 고계차분 공격에 필요한 평문의 개수가 전체 코드북의 크기보다 커지게 되어 3라운드 distinguisher의 구성이 불가능하다. 2라운드 distinguisher까지만 고계차분 공격에 이용될 수 있고 따라서 5라운드면 고계차분 공격에 안전할 것으로 판단된다.

사. Interpolation 공격

Interpolation 공격[12]은 평문, 암호문쌍을 이용하여 공격자가 작은 차수의 다항식을 구성할 수 있는 경우에 적용된다. 차수가 낮은 다항식이 구성되었으면 Lagrange 보간 공식을 이용하여 소수의 평문, 암호문쌍으로부터 계수를 결정할 수 있다.

이 공격에 안전하기 위해서는 S-box의 다항식 표현이 충분히 복잡해야 한다. ARIA의 경우 사용되는 두개의 S-box인 S와 S^{-1}는 각각 다음 식으로 표현되며 확산층과 결합되어 interpolation 공격에 대하여 충분한 내성을 갖는다고 판단된다.

$$S(x) = 39 + 55x^{127} + ccx^{191} + 3cx^{223} + aax^{239} + 0bx^{247} + d1x^{251} + e5x^{253} + 1ex^{254}$$

$$S^{-1}(x) = \frac{1}{60 + 04x + bcx^2 + f4x^4 + 71x^8 + 26x^{16} + 09x^{32} + 17x^{64} + 48x^{128}}$$

아. 키 스케줄의 취약성

취약키나 동치키의 존재 가능성은 매우 낮다. IDEA형의 취약키는 키를 암호논리의 일부로 연산에 직접 사용하였기 때문에 생기는 것으로 EXOR만으로 키를 사용하는 ARIA의 경우는 적용되지 않는다. 다른 평문에 대하여 같은 암호문을 제공하는 동치키의 경우도 발생하지 않을 것으로 판단된다. SEED의 경우 키 스케줄의 입력에 두개의 키요소를 모듈러 덧셈하기 때문에 동치키가 발생할 수 있었으나 ARIA의 경우는 라운드 함수를 Feistel 구조로 사용하므로 SEED에서와 같은 동치키는 존재하지 않는다.

ARIA의 키스케줄은 whitening부분과 라운드키 생성 부분으로 나누어 지는데 whitening 부분은 동치키, 취약키의 존재 가능성을 낮추고 라운드키의 값이 일부 노출되더라도 암호키의 정보가 얻어지기 어렵게 하는 역할을 한다. 라운드키의 생성은 whitening 부분에서 생성된 부분키로부터 비트회전을 통하여 두개 혹은 세개의 부분키를 결합하는 구조로 되어 있다. 여러 라운드 키의 일부 정보를 취합하여 부분키 정보를 획득할 수 없도록 설계되었으며 부분키의 정보가 노출되는 경우는 두개의 관련된 라운드 키의 값 256비트를 전부 찾은 경우에 한정된다. 이러한 특성으로부터 연관키 공격에 대한 내성을 갖도록 설계되었다.

5. 효율성 분석

ARIA알고리즘은 SPN 구조를 사용하면서 암호화/복호화가 동일하도록 설계되었다. 이러한 구조의 장점은 구현 시 복호화 부분을 따로 구현하지 않아도 되므로, 소프트웨어 구현에서는 코드 크기를, 하드웨어 구현에서는 게이트 수를 줄일 수 있는 장점을 가지고 있다. 또한, S-box 치환, EXOR 등 간단한 논리 만을 사용하여 구현에서 장점을 가지도록 하였다. 효율성 분석은 8비트, 32비트 소프트웨어 구현과 하드웨어 구현시 속도와 면적을 분석해 보았다.

가. 8비트 기반 구현에서의 효율성 분석

8비트 단위 구현은 사용된 연산들의 개수를 이용하여 성능을 분석할 수 있다. ARIA는 간단한 논리인 S-box 치환과 EXOR만으로 이루어져 있다. 연산의 개수는 Diffusion Layer에 의하여 주로 결정되며 ARIA의 Diffusion Layer는 각 바이트 출력에 6번의 EXOR가 사용된다. 즉, Diffusion Layer에 총 96번의 EXOR가 사용된다.

라운드에서 EXOR 연산 회수를 줄이는 다음과 같은 방법이 있다. 다음의 연산을 보자.

$$x_0 \leftarrow y_0 \oplus y_2 \oplus y_3,$$
$$x_1 \leftarrow y_1 \oplus y_2 \oplus y_3.$$

위는 4개의 EXOR가 사용되었는데, 중간 변수를 이용하면 다음과 같이 EXOR 연산 회수를 3으로 줄일 수 있다.

$$t \leftarrow y_2 \oplus y_3,$$
$$x_0 \leftarrow y_0 \oplus t,$$
$$x_1 \leftarrow y_1 \oplus t.$$

ARIA의 경우에는 24개의 중간 변수를 도입하여 Diffusion Layer의 EXOR 연산 회수를 72로 줄일 수 있다. 그러나, EXOR 연산 회수의 감소량 만큼 중간 변수가 사용되어 효율성에서 크게 차이가 나지 않거나 오히려 떨어진다.

마지막 라운드에서는 Diffusion Layer가 사용되지 않고 키 덧셈이 한번 더 사용되는 것을 고려하여 연산량을 계산하면 다음과 같다. 다음 표에서는 AES, Camellia에서의 연산량도 제시하여 비교한다. 키의 크기가 128비트인 경우 ARIA, AES, Camellia에서 주요 연산의 개수는 다음과 같다.

<표 5> ARIA, AES, Camellia에서의 주요 연산의 개수

	ARIA		AES			Camellia
테이블 참조	160	160	160	304	448	144
EXOR	1,040	824	1,040	816	608	826
Assign	160	376	160	484	160	144
Shift	0	0	576	0	0	32
If	0	0	144	0	0	0

<표 6> 8비트 기반 구현 결과(Pentium IV 2.4GHz, Pentium III 866MHz)

알고리즘	Pentium IV 2.4 GHz (Mbps) (VC++ v6.0 & gcc v3.2)	Pentium III 866 MHz (Mbps) (VC++ v6.0 & gcc v3.2)
ARIA	212	70
AES	189	65
Camellia	220	72

8비트 연산만을 사용한 코드를 Pentium IV 2.4 GHz와 Pentium III 866MHz에서 Visual C++ v6.0과 gcc v3.2를 이용하여 실행한 결과는 <표 6>과 같다. 위는 8비트 프로세서에서의 실행한 결과가 아니므로 정확한 결과는 아니다. 그러나, 컴파일러의 옵션 조절을 통하여 효율성을 측정해본 결과 세 알고리즘의 수행 속도가 크게 차이가 나지 않았다.

나. 32비트 기반 구현에서의 효율성 분석

ARIA는 128비트 단위 연산을 이용한 Diffusion Layer를 사용하지만 32비트 구현에서 효율적인 것을 선택하였다. ARIA의 Diffusion Layer A는 다음과 같은 4×4이진 행렬들을 이용하였다.

$$P_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, P_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, P_3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, P_4 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$I_4 = P_0$를 4×4 단위 행렬이라고 할 때, 다음과 같은 4×4 행렬(16×16)들을 선택한다.

$$M_1 = \begin{pmatrix} 0 & I_4 & I_4 & I_4 \\ I_4 & 0 & I_4 & I_4 \\ I_4 & I_4 & 0 & I_4 \\ I_4 & I_4 & I_4 & 0 \end{pmatrix}, M_2 = \begin{pmatrix} P_0 & 0 & 0 & 0 \\ 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 \\ 0 & 0 & 0 & P_3 \end{pmatrix}, M_3 = \begin{pmatrix} P_4 & 0 & 0 & 0 \\ 0 & P_4 & 0 & 0 \\ 0 & 0 & P_4 & 0 \\ 0 & 0 & 0 & P_4 \end{pmatrix}$$

Diffusion Layer A는 다음과 같은 위 행렬들의 곱으로 표현된다.

$$A = M_1 \cdot M_2 \cdot M_1 \cdot M_3$$

ARIA의 한 라운드는 키 덧셈, S-box 치환, Diffusion Layer로 구성되는 데, S-box 치환과 Diffusion Layer 중 M_3를 8×32테이블을 통하여 다음과 같이 구현할 수 있다.

$$P_4 \begin{pmatrix} S(x_0) \\ S(x_1) \\ S(x_2) \\ S(x_3) \end{pmatrix} = \begin{pmatrix} S(x_1) \oplus S(x_2) \oplus S(x_3) \\ S(x_0) \oplus S(x_2) \oplus S(x_3) \\ S(x_0) \oplus S(x_1) \oplus S(x_3) \\ S(x_0) \oplus S(x_1) \oplus S(x_2) \end{pmatrix}.$$

그러므로, 다음과 같은 4개의 테이블을 정의하면

$$T_0(x) = \begin{pmatrix} 0 \\ S(x) \\ S(x) \\ S(x) \end{pmatrix}, T_1(x) = \begin{pmatrix} S(x) \\ 0 \\ S(x) \\ S(x) \end{pmatrix}, T_2(x) = \begin{pmatrix} S(x) \\ S(x) \\ 0 \\ S(x) \end{pmatrix}, T_3(x) = \begin{pmatrix} S(x) \\ S(x) \\ S(x) \\ 0 \end{pmatrix}$$

$(P_4 \circ S)(x_0, x_1, x_2, x_3)^T$는 네 개의 테이블을 이용하여 $T_0(x_0) \oplus T_1(x_1) \oplus T_2(x_2) \oplus T_3(x_3)$로 계산할 수 있다. M_1은 32비트 단위 연산이고 M_2는 바이트 permuatation이므로 연산량이 적다. 즉, ARIA는 32비트 구현에 적합하도록 설계되었음을 알 수 있다. Pentium IV 2.4GHZ, Pentium III 866MHz에서 ARIA, AES, Camellia의 수행속도는 다음과 같았다.

<표 7> 32비트 기반 소프트웨어 구현 결과(Pentium IV 2.4GHz, Pentium III 866MHz)

알고리즘	Pentium IV 2.4 GHz (Mbps) (VC++ v6.0 & gcc v3.2)	Pentium III 866 MHz (Mbps) (VC++ v6.0 & gcc v3.2)
ARIA	638	177
AES	886	254
Camellia	709	190

ARIA는 32비트, 64비트 단위 연산에 기반한 Diffusion Layer를 사용하는 AES, Camellia 보다 32비트 기반 구현에서의 효율성은 다소 떨어진다. 그러나, AES, NESSIE에 제안되었던 알고리즘 중에서 ARIA보다 나은 효율성을 가지는 후보들은 서너개에 불과한 것을 볼때, ARIA는 32비트 구현 효율성도 뛰어나다고 볼 수 있다.

다. 하드웨어 구현에서의 효율성 분석

ARIA는 하드웨어 환경에서 최적이도록 설계되었다. 하드웨어 구현에서 효율성은 처리 속도 및 Chip의 면적으로 평가한다. 처리 속도는 S-box 레이어의 수에 의하여 크게 좌우되고 Diffusion Layer에 사용된 연산 복잡도에도 영향을 받는다. 128비트 키 사이즈의 ARIA, AES, Camellia를 비교해 보면, ARIA와 AES는 같은 10개의 S-box Layer가 있어 18개의 S-box Layer가 있는 Camelia보다 처리 속도가 빠르고 ARIA의 Diffusion Layer는 EXOR만으로 구성되어 있어 유한체 연산을 사용하는 AES보다 처리 속도면에서 우수하다. Chip 면적에서 보면 Feistel 구조의 Camellia보다 SPN 구조의 ARIA, AES가 약 2배의 gate를 필요로 한다. 다음은 키 스케줄을 제외한 알고리즘 부분만을 Hynix 0.25μm 공정으로 Simulation한 결과이다.

<표 8> 하드웨어 구현 Simulation 결과 (Hynix 0.25μm 공정)

알고리즘	처리 속도(Mbps)	1라운 구현에 필요한 면적(gate)
ARIA	1,871	8,935
AES	1,839	9,088
Camellia	1,112	4,971

ARIA는 하드웨어 구현에 있어서, AES에 비하여 약간 우수한 효율성을 가지며, Camellia에 비해 처리 속도가 뛰어남을 볼 수 있다. 이런 특성을 이용하면, ARIA는 하드웨어 구현을 통하여 Smart Card, 고성능 서버 등에 활용될 수 있을 것이다.

6. 결론

본 논문에서는 블록 암호 ARIA의 설계 원칙, 구조 및 동작 방법, 안전성과 효율성을 소개하였다. ARIA는 안전성 면에서 차분 공격, 선형 공격 등의 블록 암호에 대한 여러 공격 방법에 대하여 안전하며, 효율성 면에서 8 비트 소프트웨어 환경과 하드웨어 환경에서 AES 및 Camellia와 동급 또는 그 이상의 속도로 수행된다. 구조의 단순성 그리고 안전성과 효율성 분석 결과로 볼 때 ARIA는 국가기관과 민간 간에 유통되는 중요 정보 보호에 적합한 블록 암호로 판단된다.

참고 문헌

[1] 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. In Douglas R. Stinson and Stafford Tavares, editors, *Selected Areas in Cryptography*, volume 2012 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2000.

[2] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, Berlin, Heidelberg, New York, 1993.

[3] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In Jacques Stern, editor, *Advances in Cryptology - Eurocrypt'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 12–23. Springer-Verlag, Berlin, 1999.

[4] Eli Biham, Alex Biryukov, and Adi Shamir. Miss in the middle attacks on IDEA, Khufu and Khafre. In Lars R. Knudsen, editor, *Preproceedings of Fast Software Encryption Workshop 1999*, pages 121–137, 1999.

[5] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[6] A. Biryukov, *Analysis of Involutional Ciphers: Khazad and Anubis*, In Fast Software Encryption, Proceedings FSE 2003.

[7] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher square. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.

[8] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer, 2001.

[9] FIPS PUB 197. Advanced Encryption Standard(AES). National Institute of Standards and Technology, November 2001.

[10] 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.

[11] H. Gilbert and M. Minier, *A collision attack on seven rounds of Rijndael*, Proceeding of the third AES conference, pp230–241, NIST, 2000.

[12] 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.

[13] Ju-Sung Kang, Seokhie Hong, Sangjin Lee, Okyeon Yi, Choonsik Park, and Jongin Lim. Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks. *ETRI Journal*, 23(4):158–167, 2001.

[14] Kazumaro Aoki, Masayuki Kanda. *'Search for impossible Differential of E2'*. 1999, available at http://csrc.nist.gov/CryptoToolkit/aes/round1/pubcmnts.htm

[15] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *Fast Software Encryption, Second International Workshop*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, Berlin, 1995.

[16] Xuejia Lai. Higher order derivatives and differential cryptanalysis. *Communications and Cryptography*, pages 227–233, 1994.

[17] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology - Eurocrypt'93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, Berlin, 1994.

[18] NTT Laboratories. *'Security of E2 against Truncated Differential Cryptanalysis(in progress)'*. 1999, available at http://info.isl.ntt.co.jp/e2/RelDocs/

[19] C.E. Shannon 'Communication theory of secrecy systems'. Bell Systems Technical Journal, Vol.28, pp. 656-715, 1949.

[20] David Wagner. The boomerang attack. In Lars R. Knudsen, editor, *Preproceedings of Fast Software Encryption Workshop 1999*, pages 155–169, 1999.