MBC Lab.

KONKUK UNIVERSITY

ARIA – Case Study

MBC LAB
윤상준, 허주승

Content

- **ARIA**
- 검증내용
- **Case Study**
- 향후 연구 계획

ARIA

- **ARIA**는 **Academy, Research Institute, Agency**의 약어로 학·연·관이 공통으로 개발한 정보보호의 기술
- **ARIA**는 **128-bit** 데이터 블록을 처리하는 알고리즘
- **ARIA**는**128, 192, 256 bit** 암호키를 사용
- **AIRA**는 **SEED**와 함께 전자정부의 대국민 행정 서비스용으로 보급

- 알고리즘 구조

구분	Nb(입출력 블록 크기)	Nk(입력 키 출력 크기)	Nr(라운드 수)
ARIA-128	16	16	12
ARIA-192	16	24	14
ARIA-256	16	32	16

▶ 입출력의 크기 **: 128 bit**
▶ 키 크기 **: 128, 192, 256 bit**
▶ 라운드 키 크기 **: 128 bit**
▶ 라운드 수 **:** 키 크기에 따라 **12, 14, 16bit** 라운드

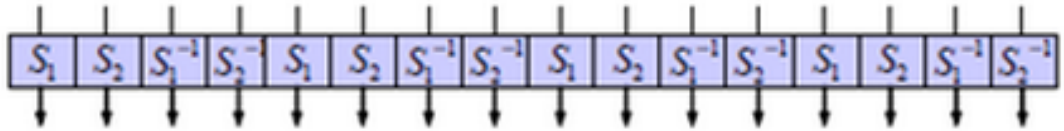
AIRA 함수

- 라운드 키 덧셈(**AddRoundKey**)
  - ▶라운드 입력 **128bit**와 함께 비트별로 **XOR** 함

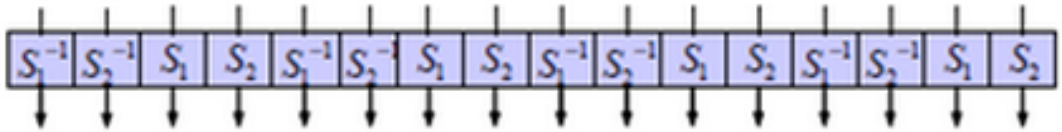
- 치환 계층(**SubstLayer**)
  - ▶**8bit** 입·출력 **S-Box**와 그들의 역 변환으로 구성 됨

- 확산 계층(**DiffLayer**)
  - ▶이진 행렬을 사용한 바이트 간의 확산 함수로 구성되어 있음

- 치환 계층(**SubstLayer**)
  - ► **S-box**에 입력되는 **8bit** 값을 **16**진법으로 표현함
  - ► **2**가지의 치환 계층을 가짐



치환 계층 (유형 1)



치환 계층 (유형 2)

- 치환 계층(SubstLayer)
  - **S-box :** 비선형 치환 테이블로 바이트 치환에 사용됨
  - **S-box**에 입력되는 **8bit** 값이 **16**진법으로 **'xy'**이면, 출력은 **'x'**행 **'y'** 열에 위치한 값이 됨.

<표 4> 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

<표 5> 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

<표 6> 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	04	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	0b	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

<표 7> 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	09	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	08	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

- 확산 계층(**DiffLayer**)
  - ▶확산 함수는 입력 **16byte**에 대하여 단위의 행렬 곱을 수행한 결과의 **16byte**를 출력으로 함

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

Scenario (암호화)

- **192bit**의 암호화

  ► **128bit** 평문

```
plaintext  : 11 11 11 11 aa aa aa aa 11 11 11 11 bb bb bb bb
```

Scenario (암호화)

- **192bit**의 암호화

```
Encryption:
 1 round : ab cc ef ff be 1a 1a c2 c2 b7 04 8e ae ea 9a 29
 2 round : d2 f1 9b ff 2f fb e4 40 53 5b 04 69 f8 6f 0f c0
 3 round : 4e 25 a5 6a 11 47 0e 73 54 ae 6b 27 6f 1f 89 40
 4 round : 88 5b d9 68 c2 b8 30 2e 9a 74 8b a8 c6 c1 b4 ef
 5 round : b5 bf dc 65 ff 30 17 11 c2 7a 26 7e f1 cd 06 d0
 6 round : 85 78 a7 6a 9f 2e d6 1a d5 33 5d db 80 cd ba 30
 7 round : 2b 96 8b b0 76 66 a5 66 85 e2 97 4f 25 85 23 2a
 8 round : 9e 74 98 be 4e 4f 41 7e c0 20 ca c0 a6 08 d2 7d
 9 round : 3a 87 b2 48 67 5d 1c c8 83 b4 44 6a 54 ea b2 b9
10 round : 9d 2c 2e 87 b2 82 2e 4f 51 58 34 0d 64 be fb d3
11 round : d6 ee 53 31 2d d3 a0 55 05 8b 82 4a 1a 41 fb 24
12 round : 11 0c 88 75 ff 30 af 0d 67 b0 07 38 16 b4 29 75
13 round : cd 14 2e a9 01 2c 0c f8 17 c6 55 dd 46 18 89 c6
14 round : 8d 14 70 62 5f 59 eb ac b0 e5 5b 53 4b 3e 46 2b
```

Scenario (복호화)

- **192bit**의 복호화

```
Decryption:
 1 round : a6 6b 56 dd 44 d6 50 aa 21 52 66 a0 e0 ab 77 eb
 2 round : ad 4c 17 49 90 81 33 02 d6 ee d8 68 ce b6 ca 13
 3 round : ff 9f 5d e0 fc 79 a4 c1 a1 a8 b1 86 e5 0e 17 47
 4 round : 6e 5c e5 bc fe f6 1c 7f 26 6b 0c 2e bd 3d b9 0b
 5 round : 39 6d 17 b9 e2 50 03 3c 42 14 37 e3 b2 df 8a 3b
 6 round : c0 aa a6 d3 ea 25 be 97 cd 0e 7d 83 ad 84 67 d8
 7 round : 7c f1 f3 fe 87 09 8c ba 7b d6 7b 90 58 ad c8 2b
 8 round : 21 08 45 2e d1 d9 d7 e1 a5 2f c1 e3 0e 18 a0 6d
 9 round : 3a 06 17 fd af d3 b4 a7 5e d9 58 3f c5 8a dc 87
10 round : 71 fd 77 18 31 f9 3a ea 8f fd 18 db ea 4e 23 ee
11 round : 3b 6b 5d 2e f1 52 5e 3d ee fe 01 df 30 7e 93 5b
12 round : d0 47 7c 66 f7 7b 81 38 7d 33 13 45 b0 dc 0c 1c
13 round : 6f 67 c8 c0 42 30 10 a6 a4 e2 1e 6b 62 42 dc 24
14 round : 11 11 11 11 aa aa aa aa 11 11 11 11 bb bb bb bb
```

검증 내용

- **CBMC**를 이용한 **ARIA** 함수 검증
  - ▶**CBMC** 변환
  - ▶각 **bit**별 **expression**수 비교
- **ARIA**의 작동 시간 체크 비교
  - ▶**OS**환경에 따른 성능 비교

Case study (1)

- **CBMC Code**
  - ► **cmbc aria128.c --function ARIA_test –program-only**

```
(3058) flag@1#56 == (!Wguard#35 ? flag@1#54 : flag@1#55)
(3059) i@1#105 == 11
(3060) Wguard#36 == !((int)(p@1#33[11]) == 0)
(3061) flag@1#57 == 1
       guard: Wguard#36
(3062) flag@1#58 == (!Wguard#36 ? flag@1#56 : flag@1#57)
(3063) i@1#106 == 12
(3064) Wguard#37 == !((int)(p@1#33[12]) == 0)
(3065) flag@1#59 == 1
       guard: Wguard#37
(3066) flag@1#60 == (!Wguard#37 ? flag@1#58 : flag@1#59)
(3067) i@1#107 == 13
(3068) Wguard#38 == !((int)(p@1#33[13]) == 0)
(3069) flag@1#61 == 1
       guard: Wguard#38
(3070) flag@1#62 == (!Wguard#38 ? flag@1#60 : flag@1#61)
(3071) i@1#108 == 14
(3072) Wguard#39 == !((int)(p@1#33[14]) == 0)
(3073) flag@1#63 == 1
       guard: Wguard#39
(3074) flag@1#64 == (!Wguard#39 ? flag@1#62 : flag@1#63)
(3075) i@1#109 == 15
(3076) Wguard#40 == !((int)(p@1#33[15]) == 0)
(3077) flag@1#65 == 1
       guard: Wguard#40
(3078) flag@1#66 == (!Wguard#40 ? flag@1#64 : flag@1#65)
(3079) i@1#110 == 16
(3080) Wguard#41 == (flag@1#66 == 1)
```

Case study (2)

- **CBMC Code – ARIA 128bit**
  - ► **cmbc aria128.c –-function ARIA_test**

```
Unwinding loop c::ARIA_test.5 iteration 6 file aria128.c line 524 function ARIA_
test
Unwinding loop c::ARIA_test.5 iteration 7 file aria128.c line 524 function ARIA_
test
Unwinding loop c::ARIA_test.5 iteration 8 file aria128.c line 524 function ARIA_
test
Unwinding loop c::ARIA_test.5 iteration 9 file aria128.c line 524 function ARIA_
test
Unwinding loop c::ARIA_test.5 iteration 10 file aria128.c line 524 function ARIA
_test
Unwinding loop c::ARIA_test.5 iteration 11 file aria128.c line 524 function ARIA
_test
Unwinding loop c::ARIA_test.5 iteration 12 file aria128.c line 524 function ARIA
_test
Unwinding loop c::ARIA_test.5 iteration 13 file aria128.c line 524 function ARIA
_test
Unwinding loop c::ARIA_test.5 iteration 14 file aria128.c line 524 function ARIA
_test
Unwinding loop c::ARIA_test.5 iteration 15 file aria128.c line 524 function ARIA
_test
Unwinding loop c::ARIA_test.5 iteration 16 file aria128.c line 524 function ARIA
_test
size of program expression: 4055 assignments
simple slicing removed 0 assignments
Generated 0 VCC(s), 0 remaining after simplification
VERIFICATION SUCCESSFUL
```

Case study (2)

- **CBMC Code – ARIA 192bit**
  - ► **cmbc aria192.c –-function ARIA_test**

```
Unwinding loop c::ARIA_test.5 iteration 6 file aria192.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 7 file aria192.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 8 file aria192.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 9 file aria192.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 10 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 11 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 12 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 13 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 14 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 15 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 16 file aria192.c line 524 function ARI
_test
size of program expression: 4428 assignments
simple slicing removed 0 assignments
Generated 0 VCC(s), 0 remaining after simplification
VERIFICATION SUCCESSFUL
```

Case study (2)

- **CBMC Code – ARIA 256bit**
  - ► **cmbc aria256.c –-function ARIA_test**

```
Unwinding loop c::ARIA_test.5 iteration 6 file aria256.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 7 file aria256.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 8 file aria256.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 9 file aria256.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 10 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 11 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 12 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 13 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 14 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 15 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 16 file aria256.c line 524 function ARI
_test
size of program expression: 4801 assignments
simple slicing removed 0 assignments
Generated 0 VCC(s), 0 remaining after simplification
VERIFICATION SUCCESSFUL
```

Case study (3)

- **CBMC Code – ARIA 128bit**
  - ► **cbmc aira128.c –-function ARIA_test –unwind 1**

```
Okay.  You were correct.

END   testing endianness.



State 39 file aria128.c line 486 function ARIA_test thread 0
------------------------------------------------------
  ARIA_test::1::i=0 (00000000000000000000000000000000)

State 41 file aria128.c line 487 function ARIA_test thread 0
------------------------------------------------------
  ARIA_test::1::mk={ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } {{ 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000 }}

State 42 file aria128.c line 486 function ARIA_test thread 0
------------------------------------------------------
  ARIA_test::1::i=1 (00000000000000000000000000000001)

Violated property:
  file aria128.c line 486 function ARIA_test
  unwinding assertion loop 0

VERIFICATION FAILED
```

Case study (3)

- **CBMC Code – ARIA 192bit**
  - ► **cbmc aira192.c –-function ARIA_test –unwind 1**

```
Okay.  You were correct.

END   testing endianness.



State 39 file aria192.c line 486 function ARIA_test thread 0
------------------------------------------------------
  ARIA_test::1::i=0 (00000000000000000000000000000000)

State 41 file aria192.c line 487 function ARIA_test thread 0
------------------------------------------------------
  ARIA_test::1::mk={ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } {{ 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000 }}

State 42 file aria192.c line 486 function ARIA_test thread 0
------------------------------------------------------
  ARIA_test::1::i=1 (00000000000000000000000000000001)

Violated property:
  file aria192.c line 486 function ARIA_test
  unwinding assertion loop 0

VERIFICATION FAILED
```

Case study (3)

- **CBMC Code – ARIA 256bit**
  - ► **cbmc aira256.c –-function ARIA_test –unwind 1**

```
Okay.  You were correct.

END   testing endianness.



State 39 file aria256.c line 486 function ARIA_test thread 0
------------------------------------------------------
  ARIA_test::1::i=0 (00000000000000000000000000000000)

State 41 file aria256.c line 487 function ARIA_test thread 0
------------------------------------------------------
  ARIA_test::1::mk={ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } {{ 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000,
 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000,
 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
 00000000, 00000000, 00000000, 00000000 }}

State 42 file aria256.c line 486 function ARIA_test thread 0
------------------------------------------------------
  ARIA_test::1::i=1 (00000000000000000000000000000001)

Violated property:
  file aria256.c line 486 function ARIA_test
  unwinding assertion loop 0

VERIFICATION FAILED
```

Case study (4)

- **CBMC Code – ARIA 128bit**
  - ► **cbmc aira128.c --function ARIA_test -- bounds-check**

```
test
Unwinding loop c::ARIA_test.5 iteration 9 file aria128.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 10 file aria128.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 11 file aria128.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 12 file aria128.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 13 file aria128.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 14 file aria128.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 15 file aria128.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 16 file aria128.c line 524 function ARI
_test
size of program expression: 4775 assignments
simple slicing removed 251 assignments
Generated 1624 VCC(s), 720 remaining after simplification
Passing problem to propositional reduction
Running propositional reduction
Solving with MiniSAT2 without simplifier
687504 variables, 8559557 clauses
empty clause: negated claim is UNSATISFIABLE, i.e., holds
Runtime decision procedure: 31.812s
VERIFICATION SUCCESSFUL
```

Case study (4)

- **CBMC Code – ARIA 192bit**
  - ► **cbmc aira192.c --function ARIA_test -- bounds-check**

```
Unwinding loop c::ARIA_test.5 iteration 9 file aria192.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 10 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 11 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 12 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 13 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 14 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 15 file aria192.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 16 file aria192.c line 524 function ARI
_test
size of program expression: 5244 assignments
simple slicing removed 251 assignments
Generated 1816 VCC(s), 816 remaining after simplification
Passing problem to propositional reduction
Running propositional reduction
Solving with MiniSAT2 without simplifier
783232 variables, 9710188 clauses
empty clause: negated claim is UNSATISFIABLE, i.e., holds
Runtime decision procedure: 37.094s
VERIFICATION SUCCESSFUL
```

Case study (4)

- **CBMC Code – ARIA 256bit**
  - ► **cbmc aira256.c --function ARIA_test -- bounds-check**

```
Unwinding loop c::ARIA_test.5 iteration 9 file aria256.c line 524 function ARIA
test
Unwinding loop c::ARIA_test.5 iteration 10 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 11 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 12 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 13 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 14 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 15 file aria256.c line 524 function ARI
_test
Unwinding loop c::ARIA_test.5 iteration 16 file aria256.c line 524 function ARI
_test
size of program expression: 5713 assignments
simple slicing removed 251 assignments
Generated 2008 VCC(s), 912 remaining after simplification
Passing problem to propositional reduction
Running propositional reduction
Solving with MiniSAT2 without simplifier
876618 variables, 10857748 clauses
empty clause: negated claim is UNSATISFIABLE, i.e., holds
Runtime decision procedure: 40.86s
VERIFICATION SUCCESSFUL
```

Case study (4)

- **SCOOT – ARIA 256bit**
  - **Make systemc_sim**
  - **Time ./systemc_sim**

```
317120835'3520251392'128171993'2718175108
1629968297'3793094657'4279455689'4069102608
698690949'1435376123'1877336613'600310695
2302384499'3121244306'3970742327'972412356
939437411'128216408'1974887620'2997986107
4048950501'3019298435'2340480092'2150627655
1964646269'837436901'4107601866'2639259484
3754769108'1033410611'2550949893'3997175353
2548299364'3206229221'1522306920'86434004
3847606758'1673794966'179411016'1828961123
2867286747'73296782'2269637538'3613767317
1440094938'1597753609'3932275159'341387058
1259938329'4263009183'510142035'3521424989
3245106737'1469437767'422124918'1215268682
2111411649'1856483220'4057646655'1491999078
2927974566'2969686177'3188488396'3458955320
1904093749'3276013016'527519892'3931741312
4040203292'3192246893'2528159517'732051385
3988475133'1997402627'748365011'604966897
2960373'897299648'164292556'3114964396
2976312705'936374495'3676004722'1543305608
80981009'3274703627'297443558'619326216
3050217290'1669619409'2118274499'2999726195
3565890413'3744799529'2480456209'2314218360
2792158424'4052746088'2719552856'2408766526
4032053029'2424712091'1100638758'2824261593
2711169716'4021560815'3810263637'1555545036
2751633354'3704984090'4140994654'2486421332
```

```
real    0m42.341s
user    0m38.277s
sys     0m0.444s
```

Next Plan

- **S-BOX**의 암호화 복호화 분리 후 **TEST**

- 다른 암호화 알고리즘 **(AES, DES, SEED)**을 동일한 방법을 통하여 비교
  - ▶각 특징 비교

- **Scoot**을 사용 비교

감사합니다.