Term Paper
Block Cipher: ARIA

Driti Singh
Divyansh Khandelwal
Tanish Gupta

Abstract :

In this Term Paper, we propose a 128-bit block cipher ARIA which is an involution substitution and permutation encryption network(SPN). We have used a 16×16 binary matrix of the maximum branch number 8 to avoid some attacks well applied to the reduced round of Rijndael in the diffusion layer.We use the same S-boxes as Rijndael to eliminate defects which are caused by a totally involution structure.
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

This Term Paper provides a complete description of ARIA. Block Cipher ARIA is an involution SPN Ciphers which has been studied by AES developers, with the following characteristics:

• ARIA accommodates key sizes of 128, 192, and 256 bits, and the block size is 128-bit long.

• ARIA uses two kinds of S-boxes and two types of substitution layers which is briefly described in further part of Term paper

• ARIA uses the same algorithm for encryption and decryption, taking advantage of its involutional diffusion matrix.

• ARIA is designed to resist many known attacks on block ciphers, including differential cryptanalysis and linear cryptanalysis.

• ARIA uses a 16 $\times$ 16 involutional binary matrix with maximum branch number of 8 as its diffusion layer.

• ARIA is designed to be efficient both in software and hardware implementation

2 Specification

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
• $\gg$ n: Right circular rotation of operand by n bits
• $\ll$ n : Left circular rotation of operand by n bits
• $\cdot$: Multiplication of two operands

2.2 Overall Structure

ARIA is a SPN block cipher with 128-, 192-, and 256-bit keys. It processes 128- bit blocks, and the number of rounds is 12, 14, and 16, depending on the key size of 128, 192, and 256 bits, respectively. The ARIA algorithm can be considered as a series of operations done to a 128-bit array called the state. The state is initialized as the plaintext input, and each operation in each round modifies the state. The final value of the state is the output of the ARIA algorithm.

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

1. **Round key addition**, where the state is XORed with a 128-bit round key.

2. **Substitution layer**, where the state goes through 16 S-boxes. There are two kinds of ARIA substitution layers, Type 1 and Type 2, and they alternate between the rounds.

3. **Diffusion layer**, where a simple 16×16 binary matrix is multiplied to the state, considered as an array of 16 bytes.

Also, there is **Key expansion operation**, where a given secret key is expanded into 13, 15, and 17 round keys, depending on the key size of 128, 192, and 256 bits, respectively.

2.3 Substitution layer

ARIA uses two types of S-boxes S_1, S_2 and their inverses S_1^{-1} , S_2^{-1} in its substitution layers. They are given in Table 1, Table 2, Table 3, and Table 4.

For example, S1(0x00) = 0x63), S1(0x05) = 0x6b), and S1(0x72) = 0x40).

ARIA has two types of substitution layers as shown.

Type 1 is used in the odd rounds, and Type 2 is used in the even rounds and this two types alternate between rounds

Table 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

Table 2: 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

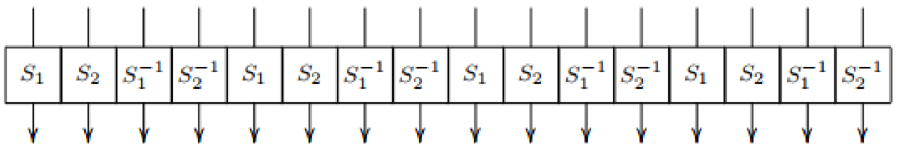
Table 3: S-box S_2

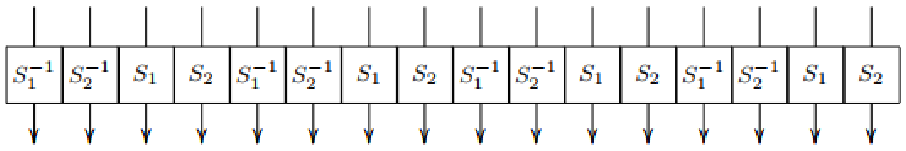
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	e2	4e	54	fc	94	c2	4a	cc	62	0d	6a	46	3c	4d	8b	d1
1	5e	fa	64	cb	b4	97	be	2b	bc	77	2e	03	d3	19	59	c1
2	1d	06	41	6b	55	f0	99	69	ea	9c	18	ae	63	df	e7	bb
3	00	73	66	fb	96	4c	85	e4	3a	09	45	aa	0f	ee	10	eb
4	2d	7f	f4	29	ac	cf	ad	91	8d	78	c8	95	f9	2f	ce	cd
5	08	7a	88	38	5c	83	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

Table 4: 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

Following are two types of substitution layer:


S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1}



2.4 Diffusion Layer :

The difiusion layer A of ARIA is a function which maps an input $(x_0, x_1, x_2...x_{15})$ of 16 bytes into an output $(y_0, y_1, y_2...y_{15})$.It is defined as follows:

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

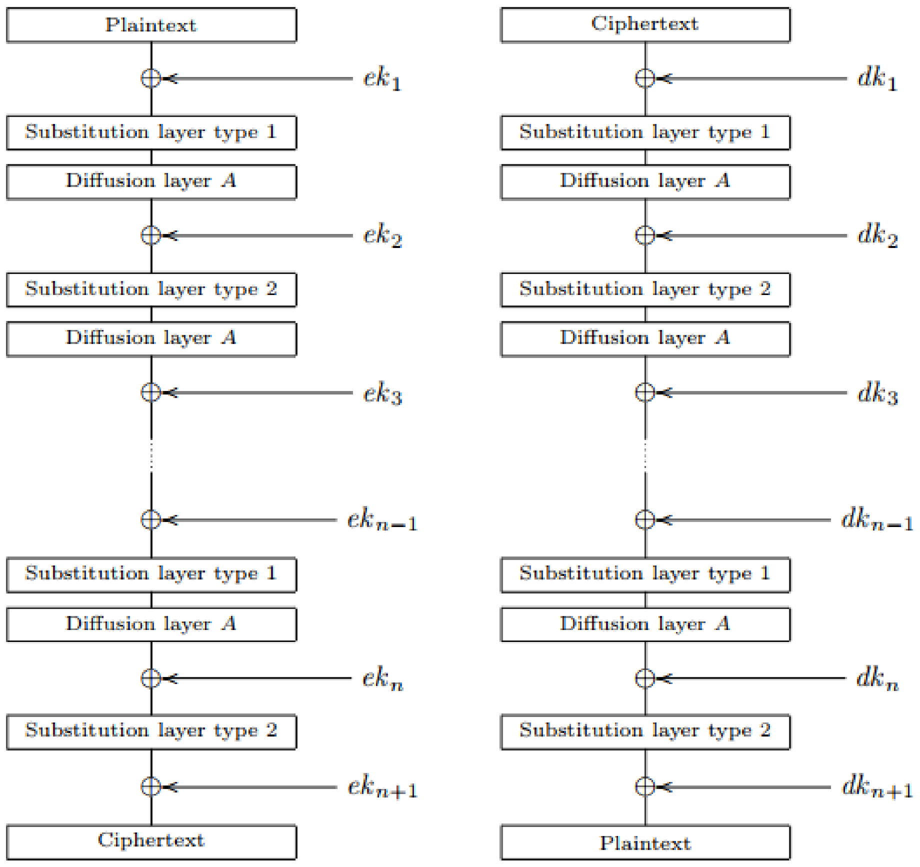
The mapping A can also be considered as a 16 * 16 binary matrix multiplication as follows:

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

The ARIA difiusion layer is designed to be an involution, i.e., for any input vector x, it satisfles x = A(A(x)).

2.5 The Cipher

The Encryption and Decryption processes of an n-round ARIA, where n is even,is shown below.



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

2.6 Key Expansion

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

Initialization >>

For initialization,four 128-bit values W_0, W_1, W_2, W_3 are generated from the master key named MK,by using a 3-Round 256-bit Fiestel Cipher.But MK can be of 128 or 192 or 256 bit.So we have to first fill out the 128 bit value KL with bits from MK and use remaining bits (if any) for the 128-bit value KR.The space remaining in the KR(if any) is to be filled with zero,so that the following equations hold.

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

Then we set

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

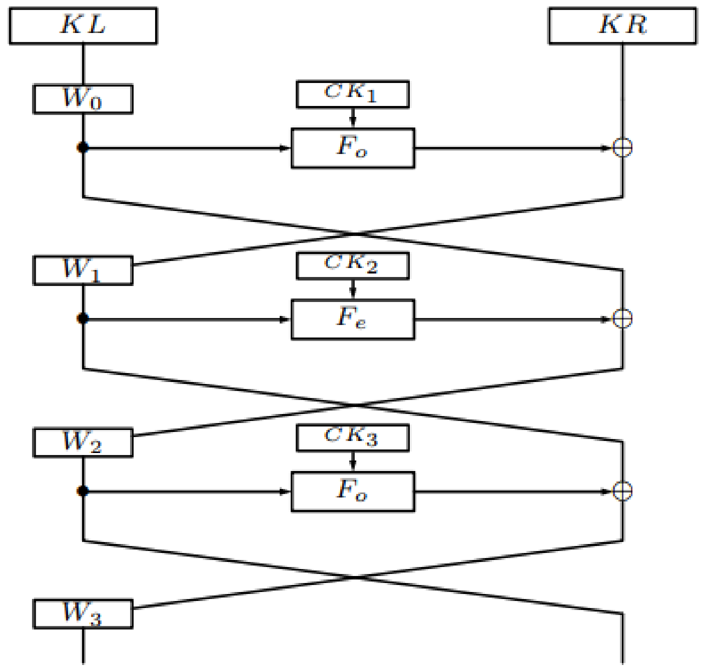
Here $F_o and F_e$ are even and odd round functions respectively,and CK_i are constants used in this process as round keys of the round functions F_o and F_e.These are calculated as follows.First,the first 128*3 bits of the fractional part of π inverse is broken into three 128 bit constants C_i.

$$\begin{aligned}
C_1 &= 0x517cc1b727220a94fe12abe8fa9a6ee0\\
C_2 &= 0x6db14acc9e21c820ff28b1d5ef5de2b0\\
C_3 &= 0xdb92371d2126e9700324977504e8c90e
\end{aligned}$$

Then the constants CK_i are defined by the following table:

Key Size	CK_1	CK_2	CK_3
128	C_1	C_2	C_3
192	C_2	C_3	C_1
256	C_3	C_1	C_2

The complete diagram of initialization process is given below:



Round key generation >>
In the round key generation,the four values W_i are combined in different ways to generate different encryption round keys ek_i and the decryption round keys df_i.

$$\begin{aligned}
ek_1 &= (W_0) \oplus (W_1 \ggg 19)\\
ek_2 &= (W_1) \oplus (W_2 \ggg 19)\\
ek_3 &= (W_2) \oplus (W_3 \ggg 19)\\
ek_4 &= (W_0 \ggg 19) \oplus (W_3)\\
ek_5 &= (W_0) \oplus (W_1 \ggg 31)\\
ek_6 &= (W_1) \oplus (W_2 \ggg 31)\\
ek_7 &= (W_2) \oplus (W_3 \ggg 31)\\
ek_8 &= (W_0 \ggg 31) \oplus (W_3)\\
ek_9 &= (W_0) \oplus (W_1 \lll 61)\\
ek_{10} &= (W_1) \oplus (W_2 \lll 61)\\
ek_{11} &= (W_2) \oplus (W_3 \lll 61)\\
ek_{12} &= (W_0 \lll 61) \oplus (W_3)\\
ek_{13} &= (W_0) \oplus (W_1 \lll 31)\\
ek_{14} &= (W_1) \oplus (W_2 \lll 31)\\
ek_{15} &= (W_2) \oplus (W_3 \lll 31)\\
ek_{16} &= (W_0 \lll 31) \oplus (W_3)\\
ek_{17} &= (W_0) \oplus (W_1 \lll 19)
\end{aligned}$$

An important note here is that the number of rounds we use are 12,14 or 16 corresponding to the key size 128,192 or 256 of the master key respectively.Since there is one more key addition layer after the last round,the number of round keys we need is 13,15 or 17.

The decryption round keys are different from encryption round keys but are derived from the encryption round keys only.The ordering of the round keys are reversed followed by the output of diffusion layer A to all the round keys except for the first and the last.

Following are the formulas of decryption keys.

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

3 Cryptanalysis

Some useful Notations and definations:

1. U is subspace of $\mathbb{F}_{2^n}$ and $U^{\perp}$ being its orthogonal subspace.
2. The symbols X_i, Y_i ,and Z_i denote the intermediate values before the substitution layer (SL), diffusion layer (DL), and AddRoundKey (AK) operations in the i-th round, respectively.
3. Linear approximation is given by $(\alpha \longrightarrow \beta)$ with an input mask α and an output mask β.
4. Differential is given by$(\delta \longrightarrow \Delta)$ with an input difference δ and an output difference Δ.
5. A set $\{a_i | a_i \in \mathbb{F}_{2^n}, 0 \leq i \leq 2^n - 1\}$ is active, if for any $0 \leq i < j \leq 2^n - 1, a_i \neq a_j$.
6. A set $\{a_i | a_i \in \mathbb{F}_{2^n}, 0 \leq i \leq 2^n - 1\}$ is balanced, if the sum of all element of the set is 0, that is $\Sigma_{i=0}^{2^n-1} a_i = 0$

3.1 Differential-Linear Cryptanalysis:

For a n-bit block cipher $E_r = E_{r1} \circ E_{r2}$ with $r = r_1 + r_2$ rounds, we apply an r_2 -round linear approximation $(\alpha \longrightarrow \beta)$ to E_{r2} with a bias ϵ and we apply an r_1-round differential $(\delta \longrightarrow \Delta)$ to E_{r1} with probability p, $(0 < p \leq 1)$ where $\delta \cdot \Delta = 0$ theh we have,

$$\begin{aligned}\beta \cdot (E_r(x) \oplus E_r(x \oplus \delta)) = {} & \alpha \cdot (E_{r1}(x \oplus \delta) \oplus E_{r1}(x)) \\ & \oplus \alpha \cdot E_{r1}(x \oplus \delta) \oplus \beta \cdot E_r(x \oplus \delta) \\ & \oplus \alpha \cdot E_{r1}(x) \oplus \beta \cdot E_r(x)\end{aligned}$$

Assuming the two round functions involved behave independently and that the two inputs $E_{r1}(x)$ and $E_{r1}(x \oplus \delta)$ of E_{r2} behaves as independent inputs with respect to the linear approximation, then when $\alpha \cdot (E_{r1}(x \oplus \delta) \oplus E_{r1}(x)) = 0$, the probability is,

$$Pr(\ \beta \cdot (E_r(x) \oplus E_r(x \oplus \delta))\ ) = \tfrac{1}{2} + 2\epsilon^2 \text{ and,}$$

$$\epsilon(U, W) \ = \ \tfrac{2}{|W|} \textstyle\sum_{v \in \mathbb{F}_2^n / \{0\}} \ \epsilon(U, v) C(v, W) \qquad - (1)$$

where, $C(v, W)$ is the capacity of the multidimensional linear approximation with nonzero input mask v and all nonzero output masks w in the space W and,
$\epsilon(U, v) \ = \ Pr(U^{\perp} / \ \{0\} \rightarrow W^{\perp}) - 1/2$, which denotes the **bias of a multidimensional differential-linear approximation** as given by Celine and others in FES 2014.

For the other cases, we assume that the approximation is a random distribution and the probability is 1/2. Thus, we have the following.

$$\begin{aligned}Pr(\ \beta \cdot (E_r(x) \oplus E_r(x \oplus \delta)) \ = \ 0) \ &= \ p \times (1/2 + 2\epsilon^2) + (1 - p)/2 \\ &= \tfrac{1}{2} + 2p\epsilon^2\end{aligned}$$

If the bias is sufficiently large, the distinguisher can be used as the basis of a differential-linear attack to distinguish E_r from a random function. In general, the attack has a data complexity of $O(p^{-2}\epsilon^{-4})$

Estimating (1) requires the estimation of $2^n|U|$ shorter differentials and $2^n|W|$ linear approximations, which is clearly infeasible in real cases. To solve this, Blondeau and others suggested decomposing it into two sums with respect to a set $V \in \mathbb{F}_2^n$.
Under the following assumption, the bias of differential-linear approximation can be approximated by only considering a subspace V of $\mathbb{F}_2^n$-

Assumption 1 Given a set V, we assume that

$$\left| \frac{2}{|W|} \sum_{v \in V/\{0\}} \epsilon(U,v)C(v,W)| \leq |\epsilon(U,W) \right|$$

3.1.1 Distinguishers for Five Rounds of ARIA

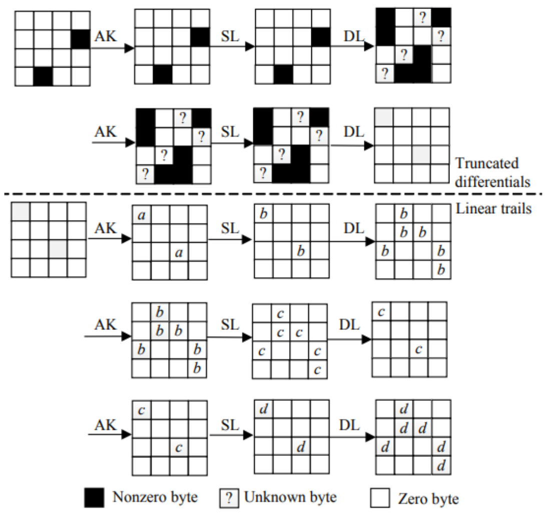
Differential-linear approximations over five rounds of ARIA, with two rounds of differentials and three rounds of linear hulls.

Constructing the Differential Characteristics The two-round differential states that given a pair of (p, p') with nonzero differences in byte 7 and byte 13, the corresponding output differentials of byte 0 and byte 10 are equal after two rounds of ARIA, that is,

$$\Delta Z_2[0] = \Delta Z_2[10]$$

as shown in the upper part of Fig 1. This can be deduced directly based on the properties of the DL layer.

Figure 1: Differential-linear approximations for five-round ARIA.



Constructing the Linear Characteristics Since the SL in the odd round is different from that in the even round, let us consider that the linear trails starts from the odd round. And let the input linear mask for the third round be $\bar{a} = (a, 0, 0, 0; 0, 0, 0, 0; 0, 0, a, 0; 0, 0, 0, 0)$, the output masks of SL for the third round be $\bar{b} = (b, 0, 0, 0; 0, 0, 0, 0; 0, 0, b, 0; 0, 0, 0, 0)$, the output masks of SL for the 4-th round be $\bar{c} = (0, 0, c, 0; c, c, 0, 0; 0, c, 0, 0; 0, 0, c, c)$, and the output masks of SL for the 5-th round be $\bar{d} = (d, 0, 0, 0; 0, 0, 0, 0; 0, 0, d, 0; 0, 0, 0, 0)$ where $a, b, c, d \in \mathbb{F}_{2^8}/\{0\}$. The square correlation of the linear hull $(\bar{a}, \bar{d})$ can be computed by,

$$\begin{aligned} C_{E_3}(\bar{a}, \bar{d}) &= \textstyle\sum_{b,c\in\mathbb{F}_{2^8}/\{0\}} Cor^2_{SL1}(\bar{a}, \bar{b})Cor^2_{SL2}(\bar{b}, \bar{c}) \times Cor^2_{SL1}(\bar{c}, \bar{d}) \\ &= \textstyle\sum_{b,c\in\mathbb{F}_{2^8}/\{0\}} Cor^2_{S1}(a, b)Cor^2_{S1}(b, a)Cor^2_{S2}(b, c) \\ &\times \textstyle\sum_{b,c\in\mathbb{F}_{2^8}/\{0\}} Cor^2_{S2}(c, b)Cor^2_{S1}(c, b)Cor^2_{S2}(b, c) \\ &\times \textstyle\sum_{b,c\in\mathbb{F}_{2^8}/\{0\}} Cor^2_{S1}(c, d)Cor^2_{S1}(d, c) \end{aligned}$$

Using the computer algorithm, for any $(\bar{a}, \bar{d})$, we get,

$C_{E_3}(\bar{a}, \bar{d}) \approx 2^{-61.7}$, $\sum_{d\in\mathbb{F}_{2^n}\{0\}} C_{E_3}(\bar{a}, \bar{d}) \approx 2^{-53.7}$ and $\sum_{a,d\in\mathbb{F}_{2^n}\{0\}} C_{E_3}(\bar{a}, \bar{d}) = 2^{-45.9}$

Let $U^{\perp} = (0, 0, 0, 0; 0, 0, 0, *; 0, 0, 0, 0; 0, *, 0, 0)$, $W = (d, 0, 0, 0; 0, 0, 0, 0; 0, 0, d, 0; 0, 0, 0, 0)$, and $V = (a, 0, 0, 0; 0, 0, 0, 0; 0, 0, a, 0; 0, 0, 0, 0)$, where $d, a \in \mathbb{F}_{2^n}\{0\}$ and * denotes a nonzero byte then we have,

$$\textstyle\sum_{v\in V/\{0\}} \epsilon(U, v)C(v, W) = 2^{-46.9}$$

By (1) and Assumption 1, to distinguish five rounds of ARIA from random permutations, the required data complexity is about,

$$\frac{|2^8|-1}{2(2^{16}-1)} 2^{93.8} \approx 2^{84.8}$$

3.2 Integral Cryptanalysis:

3.2.1 2.5-Round Integral Distinguishers of ARIA:

Assuming the input of ARIA to be B = $(B_0, B_1, ..., B_{15})$, the i-th round key be ki = $k_{i,0}, k_{i,1}, ..., k_{i,15})$, and the outputs of S-Box layer and P layer of the i-th round be $X_i = (X_{i,0}, Xi, 1, ..., Z_{i,15})$ and $Y_i = (Y_{i,0}, Y_{i,1}, ..., Y_{i,15})$,, respectively.
Now, If B_0 takes all values of $\mathbb{F}_{2^8}$ and other B_{is} are constants, then $X_{3,6}, X_{3,9}$, and $X_{3,15}$ are balanced.

Let's see it mathematically,

Input, B=$\begin{pmatrix} a & C & C & C \\ C & C & C & C \\ C & C & C & C \\ C & C & C & C \end{pmatrix}$, where C denote some constant value but not necessarily equal to each other at different positions.
and A= $S_1(a \oplus k_{1,0})$, then according to the definition of ARIA, the output of the first round is,

$$Y_1 = \begin{pmatrix} C & A\oplus\beta_4 & A\oplus\beta_8 & C \\ C & C & A\oplus\beta_9 & A\oplus\beta_{13} \\ C & A\oplus\beta_6 & C & A\oplus\beta_{14} \\ A\oplus\beta_3 & C & C & C \end{pmatrix}$$

Let $\gamma_i = \beta_i \oplus k_{2,i}$, then

$$X_2 = \begin{pmatrix} C & S_1^{-1}(A\oplus\gamma_4) & S_1^{-1}(A\oplus\gamma_8) & C \\ C & C & S_2^{-1}(A\oplus\gamma_9) & S_2^{-1}(A\oplus\gamma_{13}) \\ C & S_1(A\oplus\gamma_6) & C & S_1(A\oplus\gamma_{14}) \\ S_2(A\oplus\gamma_3) & C & C & C \end{pmatrix}$$

thus,

$$Y_{2,6} = S_2^{-1}(A\oplus\gamma_9) \oplus S_2^{-1}(A\oplus\gamma_{13}) \oplus C_1$$
$$Y_{2,9} = S_1(A\oplus\gamma_6) \oplus S_1(A\oplus\gamma_{14}) \oplus C_2$$
$$Y_{2,15} = S_1^{-1}(A\oplus\gamma_4) \oplus S_1^{-1}(A\oplus\gamma_8) \oplus C_3$$

where C_i are some constants. Now let's take $Y_{2,6}$ as an example. If $\gamma_9 = \gamma_{13}$, then $Y_{2,6} = C_1$; if $\gamma_9 \neq \gamma_{13}$ then, $S_2^{-1}(A\oplus\gamma_9) \oplus S_2^{-1}(A\oplus\gamma_{13}) \oplus C_1 = S_2^{-1}(A^*\oplus\gamma_9) \oplus S_2^{-1}(A^*\oplus\gamma_{13}) \oplus C_1$, where $A^* = A\oplus\gamma_9\oplus\gamma_{13} \neq A$.
In both cases, each value of $Y_{2,6}$ appears even times. Thus each value of $Z_{3,6}$ appears even times, which implies that $Z_{3,6}$ is balanced. This ends our proof.

The above distinguisher is simply denoted by [0,(6, 9, 15)]. Table 1 lists all possible values for [a,(b, c, d)] which means that if only the a-th byte of input takes all values of $\mathbb{F}_{2^8}$ and other bytes are constants, then $Z_{3,b}$ $Z_{3,c}$ and $Z_{3,d}$ are balanced:

Table 1. 2.5-Round Integral Distinguishers of ARIA

Active byte	Balanced bytes
0	6, 9, 15
1	7, 8, 14
2	4, 11, 13
3	5, 10, 12
4	2, 11, 13
5	3, 10, 12
6	0, 9, 15
7	1, 8, 14
8	1, 7, 14
9	0, 6, 15
10	3, 5, 12
11	2, 4, 13
12	3, 5, 10
13	2, 4, 11
14	1, 7, 8
15	0, 6, 9

3.3 3-Round Integral Distinguisher of ARIA:

Assuming the input of ARIA to be B = $(B_0, B_1, ..., B_{15})$, the i-th round key be ki = $k_{i,0}, k_{i,1}, ..., k_{i,15})$, and the outputs of S-Box layer and P layer of the i-th round be $X_i = (X_{i,0}, Xi, 1, ..., Z_{i,15})$ and $Y_i = (Y_{i,0}, Y_{i,1}, ..., Y_{i,15})$,, respectively.
Now, If B_0, B_5, B_8 takes all values of $\mathbb{F}_{2^8}^3$ and B_is are constants where $i \neq 0, 5, 8$, then $Y_{3,2}, Y_{3,5}, Y_{3,11}$ and $Y_{3,12}$ are balanced.
To prove this, 2.5 round Integral distinguisher is extended.

3.4 Multiset/Collision Attacks:

Aria,whose design has many similarities with Rijndael, seems to provide more resistance to the attacks which are considered best attacks on Rijndael like- multiset attacks and collision attacks. Aria's diffusion layer has a branch number of 8 (instead of 5 for Rijndael), and this affects the multiset and collision attacks in two ways-

- The faster diffusion does not allow a 3-round distinguisher as in Rijndael.
- Extending the (2- round) distinguisher at the top and the bottom requires much more key bytes to be guessed.

Therefore, it seems unlikely that a classical multiset attack would be able to cover more than 5 or 6 rounds depending on the key size.

3.5 Slide Attacks:

The slide attack works by analyzing the key schedule and exploiting weaknesses in it to break the cipher. The most common one is the keys repeating in a cyclic manner.
Regular slide attacks do not seem to apply to Aria because of the

irregular structure of the key schedule. Twisted slide attack are not likely to be efficient either because it would require at least two round keys to be equal, which is very unlikely given the size of Aria's round keys.

4 Implementation

8-bit software implementation >>

On an 8-bit processor ,all operations except for S-Box substition are XOR's because the implementation of S-Box requires four tables of 256 Bytes.A code for one-round except S-Box substition code computes 112 XOR's which can be reduced to 76 XOR's using four additional variables say T_0,T_1,T_2,T_3 as follows,

Thus if it is implemented serially,then only one extra 8 bit variable will be required.

32-bit software implementation >>

The ARIA software implementation mainly consists of S-Box and diffusion layer implementation.In most the cases lookup tables are adequate and efficient implementation for S-boxes,thus we need to take care about efficient implementation of diffusion layer.

Rijndael can be efficiently implemented in 32-bit processors by combining S-box and Substitution layer by 8*32 lookup tables.But this technique is suitable for a diffusion layer which is based on 32-bits.Since diffusion layer of ARIA is 128 bits,it looks to be less efficient that Rijndael.So we choose 16*16 matrices which can be implemented on 32-bit processors.

The original description of the diffusion function A takes 96 XOR operations in total.In case of 32-bit,we can compute A more efficiently by extending the lookup table for the substitution layer to some part of the diffusion layer A.

The ARIA diffusion layer A can be represented as the following matrix product:
$A = M_1 * P * M_1 * M$

where

$$M_1 = \begin{pmatrix} I & I & I & 0 \\ I & 0 & I & I \\ I & I & 0 & I \\ 0 & I & I & I \end{pmatrix}, P = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 \\ 0 & 0 & 0 & P_3 \end{pmatrix}, M = \begin{pmatrix} P_4 & 0 & 0 & 0 \\ 0 & P_4 & 0 & 0 \\ 0 & 0 & P_4 & 0 \\ 0 & 0 & 0 & P_4 \end{pmatrix}$$

The submatrices I and P_i are all 4*4.I is identity matrix and P_i are given below.

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

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

Now if we define $L = (S_1, S_2, S_1^{-1}, S_2^{-1})$ and $L^{-1} = (S_1^{-1}, S_2^{-1}, S_1, S_2)$,then we can represent the ARIA substitution layer type 1 as (L, L, L, L) and type 2 as $(L^{-1}, L^{-1}, L^{-1}, L^{-1})$.Now if we may implement $P_4 * L$ and $P_4 * L^{-1}$ efficiently.then using these we may implement the substitution layer combined with the diffusion layer efficiently.And P_4*L and $P_4 * L^{-1}$ can be implemented as lookup tables.More precisely,if (x_0, x_1, x_2, x_3) is an array of four bytes,then

$$P_4 \circ L \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} S_1(x_0) \\ S_2(x_1) \\ S_1^{-1}(x_2) \\ S_2^{-1}(x_3) \end{pmatrix}$$
$$= \begin{pmatrix} 0 \oplus S_2(x_1) \oplus S_1^{-1}(x_2) \oplus S_2^{-1}(x_3) \\ S_1(x_0) \oplus 0 \oplus S_1^{-1}(x_2) \oplus S_2^{-1}(x_3) \\ S_1(x_0) \oplus S_2(x_1) \oplus 0 \oplus S_2^{-1}(x_3) \\ S_1(x_0) \oplus S_2(x_1) \oplus S_1^{-1}(x_2) \oplus 0 \end{pmatrix}$$
$$= T_0(x_0) \oplus T_1(x_1) \oplus T_2(x_2) \oplus T_3(x_3),$$

where T_i are 8*32 lookup tables as follows:

$$T_0(x) = \begin{pmatrix} 0 \\ S_1(x) \\ S_1(x) \\ S_1(x) \end{pmatrix}, T_1(x) = \begin{pmatrix} S_2(x) \\ 0 \\ S_2(x) \\ S_2(x) \end{pmatrix}, T_2(x) = \begin{pmatrix} S_1^{-1}(x) \\ S_1^{-1}(x) \\ 0 \\ S_1^{-1}(x) \end{pmatrix}, T_3(x) = \begin{pmatrix} S_2^{-1}(x) \\ S_2^{-1}(x) \\ S_2^{-1}(x) \\ 0 \end{pmatrix}$$

Similarly $P_4 * L^{-1}$ can be implemented using the following four tables:

$$T_4(x) = \begin{pmatrix} 0 \\ S_1^{-1}(x) \\ S_1^{-1}(x) \\ S_1^{-1}(x) \end{pmatrix}, T_5(x) = \begin{pmatrix} S_2^{-1}(x) \\ 0 \\ S_2^{-1}(x) \\ S_2^{-1}(x) \end{pmatrix}, T_6(x) = \begin{pmatrix} S_1(x) \\ S_1(x) \\ 0 \\ S_1(x) \end{pmatrix}, T_7(x) = \begin{pmatrix} S_2(x) \\ S_2(x) \\ S_2(x) \\ 0 \end{pmatrix}$$

But using a special property of ARIA diffusion layer,we can implement this using only four tables T_0, T_1, T_2 and T_3 without loss of efficiency.The tables T_4, T_5, T_6 and T_7 can be obtained from T_2, T_3, T_0 and T_1 by 2-byte rotation which is simply multiplying with the matrix P_2.

And in the matrix decomposition $A = M_1 * P * M_1 * M * M_1 *$ consists of simple XOR operations on 32-bit words.Therefore the order of operations of M_1 and the 2-byte rotations can be exchanged.Also we can see the following relations among I and P_i:

$$\begin{aligned} I * P_2 &= P_2 \\ P_1 * P_2 &= P_3 \\ P_2 * P_2 &= I \\ P_3 * P_2 &= P_1 \end{aligned}$$

So,in order to implement the type 2 round function $A*(L^{-1}, L^{-1}, L^{-1}, L^{-1})$,we may use the same table lookup $T_0(x_0) \oplus T_1(x_1) \oplus T_2(x_2) \oplus T_3(x_3)$ as in $A*(L, L, L, L)$,but in the decomposition $A = M_1 * P * M_1 * M$,instead of P we have to use the following matrix $P^{'}$:

$$P^{'} = \begin{pmatrix} P_2 & 0 & 0 & 0 \\ 0 & P_3 & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & P_1 \end{pmatrix}$$

Since $A = M_1 * P * M_1 * M$ and M is a block diagonal matrix, M * S can be implemented by 8*32 lookup table. Also M_1 matrix is implemented by simple 32-bit word operations.The matrix P is a byte-oriented matrix operation that is done within each 32-bit word. Hence all these leads to a way to implement ARIA on a 32-bit based machine.On 32-bit processors, the encryption speed of ARIA is at least 70 percent of that of Rijndael and thus we can say that ARIA is also efficient on 32-bit processors.

5 Observations

1. **Impossible to calculate the encryption key from partial information of round keys**- The round keys in ARIA are always applied using the XOR operation, thus one cannot find a weak key class. The key schedule of ARIA has sufficient number of nonlinear components so as to avoid the related key attack and other such attacks.

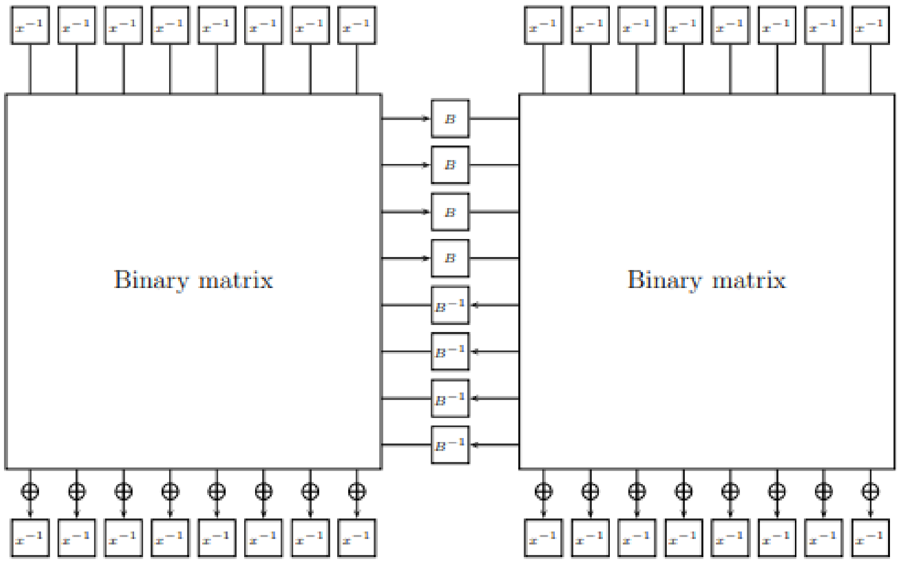
2. The minimum number of active S-boxes with respect to differential and linear cryptanalysis in r-rounds is-

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

3. Due to the similarity with Rijndael it is likely that Aria will inherit several strong properties of Rijndael which is definitely a good thing but, on the other hand we can figure out that if any serious weakness would be discovered in Rijndael, then it would apply to Aria as well. Like, one such potential threat would be the possibility of an algebraic attack of some kind. This means that Aria could be a suitable alternative to Rijndael, but may be less suited if one would only want to use it as a backup algorithm.

4. Aria has very specific features (involutional diffusion, special structure of the matrix), thus many new attack methods can be introduce over here. However, in our term paper , we have not been able to find a way to extend this to such kind of attack method .

6 Brownie Point



One interesting observation that we have choosen as our brownie point is the spliting of this cipher:

So basically this cipher can be split into two parts with particularly simple representations . The construction starts from the observation that the left half of the output of the diffusion layer (y0, . . . , y7) depends on only four independent linear combinations of the right half of the input (x8, . . . , x15).
This follows immediately from the fact that the upper right quarter of the binary diffusion matrix has rank 4. The same holds for the interaction between the right half of the output and the left half of the input.

This property allows the linear diffusion layer to be split into two binary parts where all interaction between the parts goes through 4 bytes only (in both directions).

7 Conclusion

ARIA is a simple and elegant algorithm with very much similarity to the AES-Rijndael and it is based on involtion SPN network with the following main 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.

• uses an involutional structure for efficient implementations on various environments.

Thus ,We think that ARIA is suitable for most platforms and can be widely used