Specification of **ARIA**

January, 2005

Contents

List of Figures

List of Tables

1 Introduction

This document provides a complete description of ARIA. ARIA is a block cipher with the following characteristics:

- ARIA accommodates key sizes of 128, 192, and 256 bits, and the block size is 128-bit long.
- ARIA uses a 16×16 involutional binary matrix with maximum branch number of 8 as its diffusion layer.
- 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 is designed to be efficient both in software and hardware implementations.

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
- $\ggg n$: Right circular rotation of operand by n bits
- $\lll n$: Left circular rotation of operand by n bits
- $\cdot$: Multiplication of two operands

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. Most of the operations of ARIA

are byte-oriented, therefore sometimes the state is considered as an array of 16 bytes.

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. This is because, in the last round, instead of the diffusion layer, there is another key addition.

Figure 1 illustrates the encryption and decryption processes of ARIA, where $n = 12$, 14, or 16.

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, $S_1(\texttt{0x00}) = \texttt{0x63}$), $S_1(\texttt{0x05}) = \texttt{0x6b}$), and $S_1(\texttt{0x72}) = \texttt{0x40}$).

ARIA has two types of substitution layers as shown in Figure 2 and Figure 3. The two types alternate between rounds. Type 1 is used in the odd rounds, and type 2 is used in the even 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

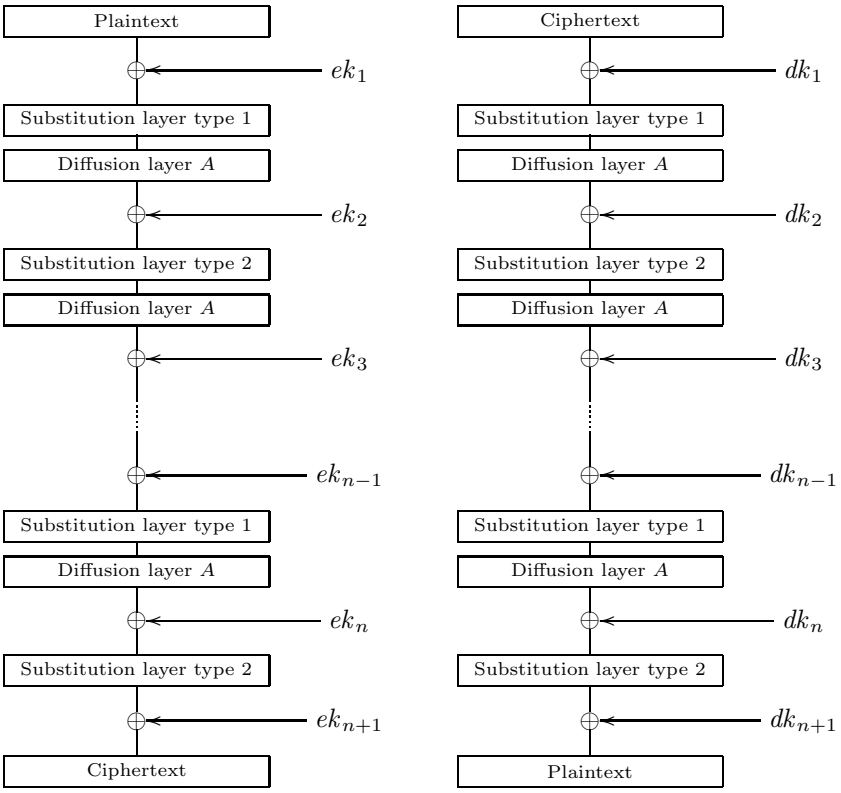

Figure 1: Encryption and decryption processes

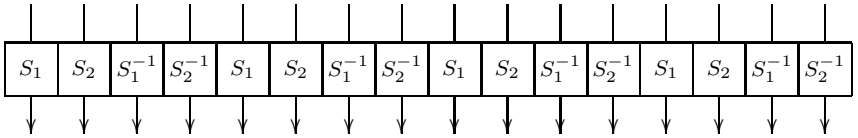

Figure 2: ARIA substitution layer, type 1

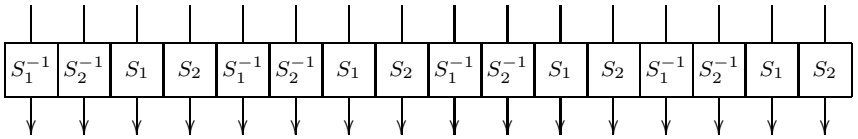

Figure 3: ARIA substitution layer, type 2

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

2.4 Diffusion layer

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

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

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 diffusion layer A is designed to be an involution, i.e., for any input vector x, it satisfies $x = A(A(x))$.

2.5 Key expansion

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

2.5.1 Initialization

In the initialization part, four 128-bit values W_0, W_1, W_2, and 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. 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, so that the following will hold.

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

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 subsections. And CK_i are constants used in this process as round keys of the round functions F_o and F_e. They are given as follows. First, the first 128×3 bits of the fractional part of π^{-1} is broken into three 128-bit constants C_i:

$$\begin{aligned} C_1 &= \texttt{0x517cc1b727220a94fe12abe8fa9a6ee0} \\ C_2 &= \texttt{0x6db14acc9e21c820ff28b1d5ef5de2b0} \\ C_3 &= \texttt{0xdb92371d2126e970324977504e8c90e0} \end{aligned}$$

Then the constants CK_i are defined by the following table:

Table 5: Constants CK_i depending on the key size

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

This initialization process is given in Figure 4.

2.5.2 Round key generation

In the round key generation, the four values W_i are combined to generate the encryption round keys ek_i and the decryption round keys dk_i.

The following is the definition for ek_i.

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

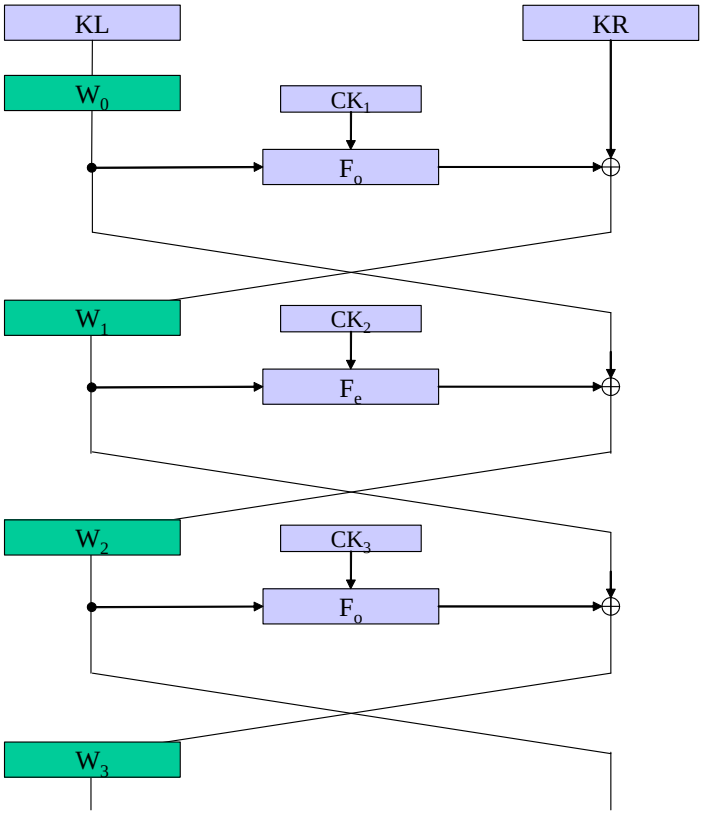

Figure 4: Initialization for ARIA key expansion

$$ek_{13} = (W_0) \oplus (W_1^{\ggg 31}), \qquad ek_{14} = (W_1) \oplus (W_2^{\ggg 31}),$$
$$ek_{15} = (W_2) \oplus (W_3^{\ggg 31}), \qquad ek_{16} = (W_0^{\ggg 31}) \oplus (W_3),$$
$$ek_{17} = (W_0) \oplus (W_1^{\ggg 19})$$

Note 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, in fact the number of round keys need is 13, 15, or 17.

The decryption round keys are different from the encryption round keys, and 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 gives the definition of dk_i.

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

A History

ARIA version 0.8 was first announced at an annual conference on security in Korea. Originally, ARIA had 10/12/14 rounds for key sizes of 128/192/256, respectively, and only two kinds of S-boxes were used.

ARIA version 0.9 was announced at ICISC 2003. In this version, four kinds of S-boxes were used.

ARIA version 1.0, the current one, was announced and distributed on its official website at `http://www.nsri.re.kr/ARIA/` in mid 2004. The number of rounds was increased to 12/14/16, and there were some modifications to the key expansion.

In December 2004, ATS(Agency for Technology and Standards) of Korea standardized ARIA as '128-bit Block Encryption Algorithm ARIA (KS X 1213)'.

B Implementation issues

B.1 32-bit software implementation

The ARIA software implementation mainly consists of S-box and diffusion layer implementation. Since in most cases lookup tables are adequate and very efficient implementation for S-boxes, the most important factor for efficient implementation is the diffusion layer.

The original description of the diffusion function A in Subsection 2.4 takes 96 XOR operations in total. In case of software implementation on 8-bit architectures, one can compute A more efficiently by introducing some intermediate variables which hold values commonly appearing in more than one output values.

In case of 32-bit software environments, we can do better by extending the lookup table for the substitution layer to some part of the diffusion layer A. Similar techniques are being used for 32-bit software implementations of other block ciphers such as AES and Camellia. We will explain this method in this subsection.

The ARIA diffusion layer A can be represented as the following matrix product:

$$A = M_1 \cdot P \cdot M_1 \cdot M,$$

where

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

The submatrices I, and P_i are all 4×4. I is the 4×4 identity matrix, and P_i are given as follows.

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

Let's 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 $(L^{-1}, L^{-1}, L^{-1}, L^{-1})$. Note that, if we may implement $P_4 \circ L$ and $P_4 \circ L^{-1}$ efficiently, then using these we may implement the substitution layer combined with the diffusion layer efficiently. And $P_4 \circ L$ and $P_4 \circ L^{-1}$ can be implemented as lookup tables. More precisely, if (x_0, x_1, x_2, x_3) is an array of four bytes, then

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

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, $P_4 \circ L^{-1}$ can be implemented using the following four tables:

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

In fact, 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. Note that 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 the matrix P_2.

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

$$I \cdot P_2 = P_2, \quad P_1 \cdot P_2 = P_3, \quad P_2 \cdot P_2 = I, \quad P_3 \cdot P_2 = P_1.$$

Therefore, in order to implement the type 2 round function $A \cdot (L^{-1}, L^{-1}, L^{-1}, L^{-1})$, 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 $A \cdot (L, L, L, L)$, but in the decomposition $A = M_1 \cdot P \cdot M_1 \cdot M$, instead of P we have to use the following matrix P':

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

B.2 Notes on hardware implementation

The hardware efficiency 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 AES and smaller than that of Camellia, throughput is almost same as that of AES and faster than that of Camellia.

Also, since ARIA is involutional, unlike AES, it requires only one procedure to be implemented for the encryption and the decryption. Therefore the area of hardware implementation for ARIA can be smaller than that of AES.

C Test vectors

In this section some test vectors for round key generation and encryption/decryption of ARIA are given. Each type of example is given for each key sizes of 128, 192, and 256. The following conventions are used to denote the state of ARIA in various operations.

- `input`: plaintext input
- `start`: input to the round function
- `key_add`: the state after XORing the round input and the round key
- `s_box`: the state after the substition layer
- `diff_lay(er)`: the state after the diffusion layer
- `output`: ciphertext output

C.1 Round key generation example for a 128-bit key

In this subsection, an example of round key generation is given for the following key.

key: `000102030405060708090a0b0c0d0e0f`

Table 6: Four values used in round key generation: 128-bit case

Variable	Value
W_0	000102030405060708090a0b0c0d0e0f
W_1	2afbea741e1746dd55c63ba1afcea0a5
W_2	7c8578018bb127e02dfe4e78c288e33c
W_3	6785b52b74da46bf181054082763ff6d

Table 7: Intermediate values for W_i: 128-bit case

Round	State	Value
1	key_round[1].input	000102030405060708090a0b0c0d0e0f
	key_round[1].key_add	517dc3b423270c93f61aa1e3f69760ef
	key_round[1].s_box	d1b933142669815c422ef194426590d3
	key_round[1].diff_layer	2afbea741e1746dd55c63ba1afcea0a5
2	key_round[2].start	2afbea741e1746dd55c63ba1afcea0a5
	key_round[2].key_add	474aa0b880368efdaaee8a7440934215
	key_round[2].s_box	1606e0703a6419ba623d7ec4725c2c97
	key_round[2].diff_layer	7c847a028fb421e725f74473ce85ed33
3	key_round[3].start	7c8578018bb127e02dfe4e78c288e33c
	key_round[3].key_add	a7174f1caa97ce902edad90dc6602a32
	key_round[3].s_box	5c2b92d6ac65ece03180e509b4ff956c
	key_round[3].diff_layer	4d7e5f5f6acd00624dd66fa988ad5fc8
	key_round[3].output	6785b52b74da46bf181054082763ff6d

Four values W_0, W_1, W_2, and W_3 used in the round key generation are displayed in Table 6.

The intermediate values for computing W_0, W_1, W_2, and W_3 are presented in Table 7.

Table 8 shows the encryption and decryption round keys.

Table 8: Round keys — 128-bit case

Round	Encryption key	Decryption key
1	d415a75c794b85c5e0d2a0b3cb793bf6	0f0aa16daee61bd7dfee5a599970fb35
2	369c65e4b11777ab713a3e1e6601b8f4	ccb3a0230b6dac1d53eef49d961aa57f
3	0368d4f13d14497b6529ad7ac809e7d0	60ea3252ac3ea9bc9ac78e79df20b5b5
4	c644552b549a263fb8d0b50906229eec	5794eadaece652f8a2ccbf68ee82a730
5	5f9c434951f2d2ef342787b1a781794c	468a335e49ec1db45d112aaf2109e5bf
6	afea2c0ce71db6de42a47461f4323c54	938ebbda880c6bb87fa01c97e68811a9
7	324286db44ba4db6c44ac306f2a84b2c	bfda5018ab33d14cc538ea5c81bd1011
8	7f9fa93574d842b9101a58063771eb7b	b5a90e77d5b94bb56e47af759fcfa05e
9	aab9c57731fcd213ad5677458fcfe6d4	21a6c28c5e1175a4378cd34dd3195a83
10	2f4423bb06465abada5694a19eb88459	8d726063ca2ceddc92afb45dd7db643e
11	9f8772808f5d580d810ef8ddac13abeb	27efd355eb17e90e5963c46515016f8d
12	8684946a155be77ef810744847e35fad	d000e81367819b077b0a657f6740e8e4
	0f0aa16daee61bd7dfee5a599970fb35	d415a75c794b85c5e0d2a0b3cb793bf6

C.2 Round key generation example for a 192-bit key

In this subsection, an example of round key generation is given for the following 192-bit key. We will show similar tables as in the 128-bit case.

key : `000102030405060708090a0b0c0d0e0f1011121314151617`

Table 9: Four values used in round key generation: 192-bit case

Variable	Value
W_0	`000102030405060708090a0b0c0d0e0f`
W_1	`e48c52301e91d991b649ed7bb7cde8ad`
W_2	`a356ea6cafe4869797a1b4eea56d38cc`
W_3	`e1898f2e0e626ccf1f58bd50713c93bb`

Table 10: Intermediate values for W_i: 192-bit case

Round	State	Value
1	`key_round[1].input`	`000102030405060708090a0b0c0d0e0f`
	`key_round[1].key_add`	`6db048cf9a24ce27f721bbdee350ecbf`
	`key_round[1].s_box`	`3ca7d445b855ecc66806fef9110883a5`
	`key_round[1].diff_layer`	`f49d40230a84cf86b649ed7bb7cde8ad`
2	`key_round[2].start`	`e48c52301e91d991b649ed7bb7cde8ad`
	`key_round[2].key_add`	`3f1e652d3fb730e1b56d7a0eb32521a3`
	`key_round[2].s_box`	`25c44ddf2570040bd2efda8b4b9dfd17`
	`key_round[2].diff_layer`	`a357e86fabe180909fa8bee5a96036c3`
3	`key_round[3].start`	`a356ea6cafe4869797a1b4eea56d38cc`
	`key_round[3].key_add`	`f22a2bdb88c68c0369b21f065ff7562c`
	`key_round[3].s_box`	`89180b59c427f01bf9d0cb21cff8b9d4`
	`key_round[3].diff_layer`	`0505dd1e10f3b55ea911502bc6f17b16`
	`key_round[3].output`	`e1898f2e0e626ccf1f58bd50713c93bb`

Table 11: Round keys — 192-bit case

Round	Encryption key	Decryption key
1	bd14be928e4305d5333b3cc231a278f6	a467dc0b2048d83faf3ffd3355a9ff5b
2	4395c65ac3dc4c6d269b1f8f81503c00	4dd0b9831c584a7f72e931dd8ede23f5
3	3121965d9e01475bda385705b2c736eb	b4a02c5b7e7cff4981137b76a1e8af63
4	40486f2e2e220c4fbf985c51507df23a	e58ecdefea05c868b394d7dabc7298b3
5	6f9ad358cd1da267352ab928609ed4f8	dc94b739beef8d4acd30f3fb4bbd0a19
6	ae5623a9583c0d48e980e054988e8170	518c97ed5d0bfaac5240e7f2dd6e2087
7	412fcd1b6cf798cb8b656d709bdc426c	a3cffeff0406d3f6fcce825184d8f470
8	f99393300e6068c91752b15e612e87ad	bab3d61669640fcd9fda3fd0cce02c65
9	36c83fac72fcbb12b498804d0fdf353d	9fe831feae7e081014a882cd3f3396f3
10	167864adca3c7e882223303622317 87f	9ad65f61fd36b359f684ac037cc53668
11	40bdfdc6a1c314e0eb90850b64a17555	622efcce90b0b68380d667bfeee031ed
12	0088ae6f6fe3cd0eff589d1011bc337b	a44f0daf2b4489a972d5971afdb359f2
13	0f49eecbdf21f0bad3effe5dfe4b2717	9777f3c8a450e691aed77bb2243af84e
14	b37e117bd54103e6e4ff711de6669d9b	044b0b0eccb4bdfb37747a14e7512e75
	a467dc0b2048d83faf3ffd3355a9ff5b	bd14be928e4305d5333b3cc231a278f6

C.3 Round key generation example for a 256-bit key

In this subsection, an example of round key generation is given for the following 256-bit key. We will show similar tables as in the 128-bit and 192-bit cases.

key : `000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f`

Table 12: Four values used in round key generation: 256-bit case

Variable	Value
W_0	000102030405060708090a0b0c0d0e0f
W_1	15169e6ec54aaf0c975414fead1c71f3
W_2	90ec92c2a800405af99389e9c88b4e62
W_3	b68cd7a1ba16abee905f7009a8e968a9

Table 13: Intermediate values for W_i: 256-bit case

Round	State	Value
1	key_round[1].input	000102030405060708090a0b0c0d0e0f
	key_round[1].key_add	db93351e2523ef770b2d9d7e08e5c701
	key_round[1].s_box	b957d9c43f6b61192bdf75a8305a3168
	key_round[1].diff_layer	05078c7dd15fb91b8f4d0ee5b1016fec
2	key_round[2].start	15169e6ec54aaf0c975414fead1c71f3
	key_round[2].key_add	446a5fd9e268a5986947bf1657861f13
	key_round[2].s_box	860acf6f3bb4063be45808bedabec0cb
	key_round[2].diff_layer	90ed90c1ac05465df19a83e2c486406d
3	key_round[3].start	90ec92c2a800405af99389e9c88b4e62
	key_round[3].key_add	fd5dd80e3621887a06bb383c27d6acd2
	key_round[3].s_box	54a42d62050697516f8a760ccc1caac9
	key_round[3].diff_layer	a39a49cf7f5c04e2070b64f705f5195a
	key_round[3].output	b68cd7a1ba16abee905f7009a8e968a9

Table 14: Round keys — 256-bit case

Round	Encryption key	Decryption key
1	8e3f60a1d7c8deae5de898e18e92dbac	f37728567c61bca7affc62e88395a6bb
2	7cdacc735712fa0c9f5f4bccdc2148e2	fd62e7342bfde4dbd9499dbb605b04bf
3	bdf9a41332f477182cee5be2268a7b7f	0fddde54a27fd06456f6779dcf03c84d
4	174d37a19a56cb6e309f910889a80928	a1d0cf146ae42a0d5dad9e70003bdec0
5	5a39e1e52e283ada829c541222a527f2	d8e599b24da2fa817d5093a776793345
6	840002abe4938a89c754944b5e3b6220	bfc0e457ae1820ff7e0efbbd80db257e
7	c13e4391c519ef198dbede34e835ae71	ec43f94c756c16adb2acd64b19b6c5d8
8	ae96cbbfba14afe898557c07b8fb7cbf	80f83e941cc54e3630d40048da4028d7
9	92eb809cd1a688396aabd9c6d4a45bee	fbe497edad612923021e12d063a84f41
10	4a24ef53fc5bc6c0c54986a6f81c79f8	e0fdffae2d79e75a524013b7f04991a8
11	42e77cc39d1d6d4fcf42131dffc99b1f	13e967b0fdd52624b97bedf6e9c0a883
12	578df6e0db970a2f705f5049c869c869	0ff5cf18dbcdaac866ea6cac40442506
13	62a455854faf0c785e8732f286864138	6dddbb6cd6ef2ef19466c369942c10fa
14	4116be43b9836bf87311b3cfe56a3892	fe0124176f59613ebf4fbb7dc11ae43f
15	4de7c735e02ff85e2de73dbd13cd25b2	65d35c19276918ffc078ed2e183c1f93
16	348e54a23e122eeb1659f70e28e9e9a8	772cce8c5b4055fd75160b2faf0c9165
	f37728567c61bca7affc62e88395a6bb	8e3f60a1d7c8deae5de898e18e92dbac

C.4 Encryption and Decryption example: 128-bit case

In this subsection, an example in ARIA encryption and decryption for a 128-bit key is given. In this and the following subsections for 192-bit and 256-bit keys, we will use the following plaintext.

00112233445566778899aabbccddeeff.

In this subsection, we will use the following 128-bit key:

000102030405060708090a0b0c0d0e0f.

Table 15: Encryption example for ARIA-128

Round	State	Value
1	round[0].input	00112233445566778899aabbccddeeff
	round[1].start	00112233445566778899aabbccddeeff
	round[1].key_add	d404856f3d1ee3b2684b0a0807a4d509
	round[1].s_box	489467d927594dd54595a350c5a9b539
	round[1].diff_lay	7fc7f12befd0a0791de87fa96b469f52
2	round[2].start	7fc7f12befd0a0791de87fa96b469f52
	round[2].key_add	495b94cf5ec7d7d26cd241b70d4727a6
	round[2].s_box	a4e222da9d5b0ea2b8c98334f358ccd4
	round[2].diff_lay	ac8de17e49f7c5117618993162b189e9
3	round[3].start	ac8de17e49f7c5117618993162b189e9
	round[3].key_add	afe5358f74e38c6a1331344baab86e39
	round[3].s_box	795ad99a9233f00a7d7328c3ac7045aa
	round[3].diff_lay	c3e8d59ec2e62d5249ca2741653cb7dd
4	round[4].start	c3e8d59ec2e62d5249ca2741653cb7dd
	round[4].key_add	05ac80b5967c0b6df11a9248631e2931
	round[4].s_box	3644cdf235ee2bca2bfb4f8d00c4a573
	round[4].diff_lay	5d4aebb165e141ff759f669e1e85cc45
5	round[5].start	5d4aebb165e141ff759f669e1e85cc45
	round[5].key_add	02d6a8f83413931041b8e12fb904b509
	round[5].s_box	771c6ff718cb223e8370e04d5694d239
	round[5].diff_lay	7806e469f68874c5004b5f4a046bbcfa
6	round[6].start	7806e469f68874c5004b5f4a046bbcfa
	round[6].key_add	d7ecc8651195c21b42ef2b2bf05980ae
	round[6].s_box	0d80e821e34b2503f6d3f1ae171ecd9d
	round[6].diff_lay	110f93c9a630cdd51f97d2202413345a
7	round[7].start	110f93c9a630cdd51f97d2202413345a
	round[7].key_add	234d1512e28a8063dbdd1126d6bb7f76
	round[7].s_box	262f2f5e98f13a2cb944e377f68a6bad
	round[7].diff_lay	e054428ef088fef97928241cd3be499e
8	round[8].start	e054428ef088fef97928241cd3be499e
	round[8].key_add	9fcbebbb8450bc4069327c1ae4cfa2e5
	round[8].s_box	6e13e98a4f8c652de46c102eae453a5a
	round[8].diff_lay	5734f38ea1ca3ddd102e71f95e1d5f97

Continues on next page...

Continued from previous page...

Round	State	Value
9	round[9].start	5734f38ea1ca3ddd102e71f95e1d5f97
	round[9].key_add	fd8d36f99036efcebd7806bcd1d2b943
	round[9].s_box	541f244c6085614e7a07a5183ea2dbaf
	round[9].diff_lay	4903325be3e500cccd52fba4354a39ae
10	round[10].start	4903325be3e500cccd52fba4354a39ae
	round[10].key_add	664711e0e5a35a7617046f05abf2bdf7
	round[10].s_box	d35882902ababe428787a8c20eb57af8
	round[10].diff_lay	cb8c508e2c4f87880639dc896d25ec9d
11	round[11].start	cb8c508e2c4f87880639dc896d25ec9d
	round[11].key_add	540b220ea312df8587372454c1364776
	round[11].s_box	204694620a64ef3617e4a602788516ad
	round[11].diff_lay	e7e0d2457ed73d23d481424095afdca0
12	round[12].start	e7e0d2457ed73d23d481424095afdca0
	round[12].key_add	6164462f6b8cda5d2c913608d24c830d
	round[12].s_box	d8125abb058257a4424705627f35ec4d
	round[12].output	d718fbd6ab644c739da95f3be6451778

Table 16: Decryption example for ARIA-128

Round	State	Value
1	round[0].input	d718fbd6ab644c739da95f3be6451778
	round[1].start	d718fbd6ab644c739da95f3be6451778
	round[1].key_add	d8125abb058257a4424705627f35ec4d
	round[1].s_box	6164462f6b8cda5d2c913608d24c830d
	round[1].diff_lay	ecf534410109432b440a529fee9fb3d2
2	round[2].start	ecf534410109432b440a529fee9fb3d2
	round[2].key_add	204694620a64ef3617e4a602788516ad
	round[2].s_box	540b220ea312df8587372454c1364776
	round[2].diff_lay	b3b2b0c2868417fe1d4026bbd195cf4d
3	round[3].start	b3b2b0c2868417fe1d4026bbd195cf4d
	round[3].key_add	d35882902ababe428787a8c20eb57af8
	round[3].s_box	664711e0e5a35a7617046f05abf2bdf7
	round[3].diff_lay	038bce968c6333b6d8cb1a70d0207c9f
4	round[4].start	038bce968c6333b6d8cb1a70d0207c9f
	round[4].key_add	541f244c6085614e7a07a5183ea2dbaf
	round[4].s_box	fd8d36f99036efcebd7806bcd1d2b943
	round[4].diff_lay	2899dad406607899b97d3a818f4cdfe5
5	round[5].start	2899dad406607899b97d3a818f4cdfe5
	round[5].key_add	6e13e98a4f8c652de46c102eae453a5a
	round[5].s_box	9fcbebbb8450bc4069327c1ae4cfa2e5
	round[5].diff_lay	b5a1948410fd5194c6e4ffe010027a04
6	round[6].start	b5a1948410fd5194c6e4ffe010027a04
	round[6].key_add	262f2f5e98f13a2cb944e377f68a6bad
	round[6].s_box	234d1512e28a8063dbdd1126d6bb7f76
	round[6].diff_lay	b25ab8394878f44f33eb1bf296a3dd8c
7	round[7].start	b25ab8394878f44f33eb1bf296a3dd8c
	round[7].key_add	0d80e821e34b2503f6d3f1ae171ecd9d
	round[7].s_box	d7ecc8651195c21b42ef2b2bf05980ae
	round[7].diff_lay	c2b56180cd72698bed374f38c95b7267
8	round[8].start	c2b56180cd72698bed374f38c95b7267
	round[8].key_add	771c6ff718cb223e8370e04d5694d239
	round[8].s_box	02d6a8f83413931041b8e12fb904b509
	round[8].diff_lay	17e20f7e6bff5e6e1c779cc0d3ddfff0
9	round[9].start	17e20f7e6bff5e6e1c779cc0d3ddfff0
	round[9].key_add	3644cdf235ee2bca2bfb4f8d00c4a573
	round[9].s_box	05ac80b5967c0b6df11a9248631e2931
	round[9].diff_lay	f428b9f9581f1dd6efdc9c9e7bab2194
10	round[10].start	f428b9f9581f1dd6efdc9c9e7bab2194
	round[10].key_add	795ad99a9233f00a7d7328c3ac7045aa
	round[10].s_box	afe5358f74e38c6a1331344baab86e39
	round[10].diff_lay	830df18f764ce7ace1aa4751e659a359
11	round[11].start	830df18f764ce7ace1aa4751e659a359
	round[11].key_add	a4e222da9d5b0ea2b8c98334f358ccd4
	round[11].s_box	495b94cf5ec7d7d26cd241b70d4727a6
	round[11].diff_lay	98948fca40d8d6d23e9fc62fa2e95ddd

Continues on next page...

Continued from previous page...

Round	State	Value
12	round[12].start	98948fca40d8d6d23e9fc62fa2e95ddd
	round[12].key_add	489467d927594dd54595a350c5a9b539
	round[12].s_box	d404856f3d1ee3b2684b0a0807a4d509
	round[12].output	00112233445566778899aabbccddeeff

C.5 Encryption and Decryption example: 192-bit case

In this subsection, an example in ARIA encryption and decryption for a 192-bit key is given. The same plaintext used for 128-bit example will be used, and the key is:

000102030405060708090a0b0c0d0e0f1011121314151617.

Table 17: Encryption example for ARIA-192

Round	State	Value
1	round[0].input	00112233445566778899aabbccddeeff
	round[1].start	00112233445566778899aabbccddeeff
	round[1].key_add	bd059ca1ca1663a2bba29679fd7f9609
	round[1].s_box	7ac21ce974be00d2eafd35bd54403539
	round[1].diff_lay	ff0a53eb839b686852dad8cf18de2cf2
2	round[2].start	ff0a53eb839b686852dad8cf18de2cf2
	round[2].key_add	bc9f95b1404724057441c740998e10f2
	round[2].s_box	78d72ae1725836c2ca22c62df969cae0
	round[2].diff_lay	ee4161aac78ae47750dfe66aff08763b
3	round[3].start	ee4161aac78ae47750dfe66aff08763b
	round[3].key_add	df60f7f7598ba32c8ae7b16f4dcf40d0
	round[3].s_box	9eff26eacb4f71d47ef356d9e3da72b2
	round[3].diff_lay	75619b2290bbf0fa4017e4b1b523a8c7
4	round[4].start	75619b2290bbf0fa4017e4b1b523a8c7
	round[4].key_add	3529f40cbe99fcb5ff8fb8e0e55e5afd
	round[4].s_box	d943bf3c5a26b0f27d9a6c902a10beba
	round[4].diff_lay	9f1cc55fa9d75818f408c2259ea62d2b
5	round[5].start	9f1cc55fa9d75818f408c2259ea62d2b
	round[5].key_add	f086160764cafa7fc1227b0dfe38f9d3
	round[5].s_box	8c84ff78436e144178410309bb3a691c
	round[5].diff_lay	454e7efa1988f1182fa9a316dde0f831
6	round[6].start	454e7efa1988f1182fa9a316dde0f831
	round[6].key_add	eb185d5341b4fc50c6294342456e7941
	round[6].s_box	3c2a4c38f814b008c7431af468cab67f
	round[6].diff_lay	88c39bb29e7a83334998a21901abe928
7	round[7].start	88c39bb29e7a83334998a21901abe928
	round[7].key_add	c9ec56a9f28d1bf8c2fdcf699a77ab44
	round[7].s_box	ddd6b9a4891f44f725ba5f27b8260edd
	round[7].diff_lay	deab0665aacde8aad1c1986ff9967c5e
8	round[8].start	deab0665aacde8aad1c1986ff9967c5e
	round[8].key_add	27389555a4ad8063c693293198b8fbf3
	round[8].s_box	3d532a831d46cd31c75ca573e25a0fa5
	round[8].diff_lay	9d8112c94f9e6711187b1a34487ac5e5
9	round[9].start	9d8112c94f9e6711187b1a34487ac5e5
	round[9].key_add	ab492d653d62dc03ace39a7947a5f0d8
	round[9].s_box	6278fa97270e931b913337bda0bf1790
	round[9].diff_lay	297d76553d4b984f3e8d4ed52464459d

Continues on next page...

Continued from previous page...

Round	State	Value
10	round[10].start	297d76553d4b984f3e8d4ed52464459d
	round[10].key_add	3f0512f8f777e6c71cae7ee306553de2
	round[10].s_box	25b9c98926198ef6c42bf333a524275b
	round[10].diff_lay	cd372f097eb243c8c0ba6c39b5c7ac23
11	round[11].start	cd372f097eb243c8c0ba6c39b5c7ac23
	round[11].key_add	8d8ad2cfdf7157282b2ae932d166d976
	round[11].s_box	5df17f459eabda57f118eb6c3e58e5ad
	round[11].diff_lay	55f9a19b5c91e0957b6a4f3049d7bc0c
12	round[12].start	55f9a19b5c91e0957b6a4f3049d7bc0c
	round[12].key_add	55710ff433722d9b8432d220586b8f77
	round[12].s_box	edfa763d66dcd83e4f6cb51d5e237326
	round[12].diff_lay	f0cf94f7402d0130ff23055245ef68ea
13	round[13].start	f0cf94f7402d0130ff23055245ef68ea
	round[13].key_add	ff867a3c9f0cf18a2cccfb0fbba44ffd
	round[13].s_box	1684bd0cdb3c2bbb7116633ceaa992a2
	round[13].diff_lay	a01563f5ea1626ad9ac1dcbf6931456e
14	round[14].start	a01563f5ea1626ad9ac1dcbf6931456e
	round[14].key_add	136b728e3f57254b7e3eada28f57d8f5
	round[14].s_box	8223401325933f958a9b95fd73936122
	round[14].output	26449c1805dbe7aa25a468ce263a9e79

Table 18: Decryption example for ARIA-192

Round	State	Value
1	round[0].input	26449c1805dbe7aa25a468ce263a9e79
	round[1].start	26449c1805dbe7aa25a468ce263a9e79
	round[1].key_add	8223401325933f958a9b95fd73936122
	round[1].s_box	136b728e3f57254b7e3eada28f57d8f5
	round[1].diff_lay	5b54048fc76461c403ff52e16477b157
2	round[2].start	5b54048fc76461c403ff52e16477b157
	round[2].key_add	1684bd0cdb3c2bbb7116633ceaa992a2
	round[2].s_box	ff867a3c9f0cf18a2cccfb0fbba44ffd
	round[2].diff_lay	595a5a6618a02777ce7fce6bffcbdc45
3	round[3].start	595a5a6618a02777ce7fce6bffcbdc45
	round[3].key_add	edfa763d66dcd83e4f6cb51d5e237326
	round[3].s_box	55710ff433722d9b8432d220586b8f77
	round[3].diff_lay	b87fb2aa74ae123f428c3cb6822a7d1e
4	round[4].start	b87fb2aa74ae123f428c3cb6822a7d1e
	round[4].key_add	5df17f459eabda57f118eb6c3e58e5ad
	round[4].s_box	8d8ad2cfdf7157282b2ae932d166d976
	round[4].diff_lay	f92d7eb098f603bc091b00c8ee992d42
5	round[5].start	f92d7eb098f603bc091b00c8ee992d42
	round[5].key_add	25b9c98926198ef6c42bf333a524275b
	round[5].s_box	3f0512f8f777e6c71cae7ee306553de2
	round[5].diff_lay	33f46d7a7a0569b7c373d04f7dd13717

Continues on next page...

Continued from previous page...

Round	State	Value
6	round[6].start	33f46d7a7a0569b7c373d04f7dd13717
	round[6].key_add	6278fa97270e931b913337bda0bf1790
	round[6].s_box	ab492d653d62dc03ace39a7947a5f0d8
	round[6].diff_lay	9e9cd47c19401ec73b9227226682fbd5
7	round[7].start	9e9cd47c19401ec73b9227226682fbd5
	round[7].key_add	3d532a831d46cd31c75ca573e25a0fa5
	round[7].s_box	27389555a4ad8063c693293198b8fbf3
	round[7].diff_lay	67656fb2e07b4b3aba6060f774c622b8
8	round[8].start	67656fb2e07b4b3aba6060f774c622b8
	round[8].key_add	ddd6b9a4891f44f725ba5f27b8260edd
	round[8].s_box	c9ec56a9f28d1bf8c2fdcf699a77ab44
	round[8].diff_lay	a3c27dc6566ab818d3eb983957f9208c
9	round[9].start	a3c27dc6566ab818d3eb983957f9208c
	round[9].key_add	3c2a4c38f814b008c7431af468cab67f
	round[9].s_box	eb185d5341b4fc50c6294342456e7941
	round[9].diff_lay	1652a019be58a7188ec5af0ac7ff5f74
10	round[10].start	1652a019be58a7188ec5af0ac7ff5f74
	round[10].key_add	8c84ff78436e144178410309bb3a691c
	round[10].s_box	f086160764cafa7fc1227b0dfe38f9d3
	round[10].diff_lay	bb6d43f2ca960671fd4c0b2fc4f08f57
11	round[11].start	bb6d43f2ca960671fd4c0b2fc4f08f57
	round[11].key_add	d943bf3c5a26b0f27d9a6c902a10beba
	round[11].s_box	3529f40cbe99fcb5ff8fb8e0e55e5afd
	round[11].diff_lay	3ab02b45e00bf87d0c26c1c31e692b40
12	round[12].start	3ab02b45e00bf87d0c26c1c31e692b40
	round[12].key_add	9eff26eacb4f71d47ef356d9e3da72b2
	round[12].s_box	df60f7f7598ba32c8ae7b16f4dcf40d0
	round[12].diff_lay	efa0d929d608d05364f5bd9fdd5332ae
13	round[13].start	efa0d929d608d05364f5bd9fdd5332ae
	round[13].key_add	78d72ae1725836c2ca22c62df969cae0
	round[13].s_box	bc9f95b1404724057441c740998e10f2
	round[13].diff_lay	7e8917e7b80abd29dd894fa9b3111b4c
14	round[14].start	7e8917e7b80abd29dd894fa9b3111b4c
	round[14].key_add	7ac21ce974be00d2eafd35bd54403539
	round[14].s_box	bd059ca1ca1663a2bba29679fd7f9609
	round[14].output	00112233445566778899aabbccddeeff

C.6 Encryption and Decryption example: 256-bit case

In this subsection, an example in ARIA encryption and decryption for a 256-bit key is given. The same plaintext used for 128-bit and 192-bit example will be used, and the key is:

000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f.

Table 19: Encryption example for ARIA-256

Round	State	Value
1	round[0].input	00112233445566778899aabbccddeeff
	round[1].start	00112233445566778899aabbccddeeff
	round[1].key_add	8e2e4292939db8d9d571325a424f3553
	round[1].s_box	19e7f69edc259a7b03aba1e52ccdd95f
	round[1].diff_lay	64739617aa297fe46a5146919209e915
2	round[2].start	64739617aa297fe46a5146919209e915
	round[2].key_add	18a95a64fd3b85e8f50e0d5d4e28a1f7
	round[2].s_box	34a4be36219897617762d7a4b65732f8
	round[2].diff_lay	f01c2fdb0bccbf37a8bf5f2e9c034ffb
3	round[3].start	f01c2fdb0bccbf37a8bf5f2e9c034ffb
	round[3].key_add	4de58bc83938c82f845104ccba893484
	round[3].s_box	e35ace4a123ab14d5f7a3007f4232886
	round[3].diff_lay	c7eebca8e1e6fd2e73e9f57de119d455
4	round[4].start	c7eebca8e1e6fd2e73e9f57de119d455
	round[4].key_add	d0a38b097bb036404376647568b1dd7d
	round[4].s_box	60ba3d0d03eb052d64ad4356f76fc1b9
	round[4].diff_lay	6c7ce71dfc22061861546c85a30dbff1
5	round[5].start	6c7ce71dfc22061861546c85a30dbff1
	round[5].key_add	364506f8d20a3cc2e3c8389781a89803
	round[5].s_box	05cfa5f7b56a6d05113576150c7ee21b
	round[5].diff_lay	97f963953737942369362139 2e83efc9
6	round[6].start	97f963953737942369362139 2e83efc9
	round[6].key_add	13f9613ed3a41eaaae62b57270b88de9
	round[6].s_box	824cef10a95d7239be08d50cd05a5da1
	round[6].diff_lay	7a4c3f387ed40316706095ea3ce83f9d
7	round[7].start	7a4c3f387ed40316706095ea3ce83f9d
	round[7].key_add	bb727ca9bbcdec0ffdde4bded4dd91ec
	round[7].s_box	ea0c01a4ea82833c549bccf94844ac80
	round[7].diff_lay	eab89889e8398c8a38fa340c982de87d
8	round[8].start	eab89889e8398c8a38fa340c982de87d
	round[8].key_add	442e5336522d2362a0af480b20d694c2
	round[8].s_box	861aed854840260e47fc524654ec2251
	round[8].diff_lay	9e1d6512590a730035caf4a431d293bb
9	round[9].start	9e1d6512590a730035caf4a431d293bb
	round[9].key_add	0cf6e58e88acfb395f612d62e576c855
	round[9].s_box	feb32a69c4fe63aacf87fa08d942b124
	round[9].diff_lay	75cb1bab78f698e5bfb077c2062993b2

Continues on next page...

Continued from previous page...

Round	State	Value
10	round[10].start	75cb1bab78f698e5bfb077c2062993b2
	round[10].key_add	3feff4f884ad5e257af9f164fe35ea4a
	round[10].s_box	25d3bf894f4658f0bd4ca1360cc887c8
	round[10].diff_lay	203c974b18b7434de855954eb2d058b1
11	round[11].start	203c974b18b7434de855954eb2d058b1
	round[11].key_add	62dbeb8885aa2e02271786534d19c3ae
	round[11].s_box	aa0a3c529739c399cc2bdc5fe377332b
	round[11].diff_lay	a5b315cd24206c9c2ed5049bfbc511a3
12	round[12].start	a5b315cd24206c9c2ed5049bfbc511a3
	round[12].key_add	f23ee32dffb766b35e8a54d233acd9ca
	round[12].s_box	049b11df7d7033f59dbb20a26644356e
	round[12].diff_lay	c6ba5f7201f9596a1d2d3aae33e4fa54
13	round[13].start	c6ba5f7201f9596a1d2d3aae33e4fa54
	round[13].key_add	a41e0af74e56551243aa085cb562bb6c
	round[13].s_box	4959a3ea2f2aed5e1a39bf54d50efedf
	round[13].diff_lay	fbfe7a26af3be9cb0fa8452af1bb3585
14	round[14].start	fbfe7a26af3be9cb0fa8452af1bb3585
	round[14].key_add	bae8c46516b882337cb9f6e514d10d17
	round[14].s_box	c09c1c21ff5a13fb017d425a9b0fd72b
	round[14].diff_lay	6971d8a121818c613e035108d445b44d
15	round[15].start	6971d8a121818c613e035108d445b44d
	round[15].key_add	24961f94c1ae743f13e46cb5c78891ff
	round[15].s_box	367bcb04789dcaed7d7db8fec649ac60
	round[15].diff_lay	531d29e52f0e5ab9498ecc4dd299f8f0
16	round[16].start	531d29e52f0e5ab9498ecc4dd299f8f0
	round[16].key_add	67937d47111c74525fd73b43fa701158
	round[16].s_box	0a5cff91e3d692888473e22914b88247
	round[16].output	f92bd7c79fb72e2f2b8f80c1972d24fc

Table 20: Decryption example for ARIA-256

Round	State	Value
	round[0].input	f92bd7c79fb72e2f2b8f80c1972d24fc
1	round[1].start	f92bd7c79fb72e2f2b8f80c1972d24fc
	round[1].key_add	0a5cff91e3d692888473e22914b88247
	round[1].s_box	67937d47111c74525fd73b43fa701158
	round[1].diff_lay	cb192c3053602e36a4342545a612a8df
2	round[2].start	cb192c3053602e36a4342545a612a8df
	round[2].key_add	367bcb04789dcaed7d7db8fec649ac60
	round[2].s_box	24961f94c1ae743f13e46cb5c78891ff
	round[2].diff_lay	cf41c2755d25c39f578b35c7540c1f66
3	round[3].start	cf41c2755d25c39f578b35c7540c1f66
	round[3].key_add	c09c1c21ff5a13fb017d425a9b0fd72b
	round[3].s_box	bae8c46516b882337cb9f6e514d10d17
	round[3].diff_lay	e8896cfe45cec75347942124d535201f

Continues on next page...

Continued from previous page...

Round	State	Value
4	round[4].start	e8896cfe45cec75347942124d535201f
	round[4].key_add	4959a3ea2f2aed5e1a39bf54d50efedf
	round[4].s_box	a41e0af74e56551243aa085cb562bb6c
	round[4].diff_lay	dc7e886d30d2c974e0ebb305103d062b
5	round[5].start	dc7e886d30d2c974e0ebb305103d062b
	round[5].key_add	049b11df7d7033f59dbb20a26644356e
	round[5].s_box	f23ee32dffb766b35e8a54d233acd9ca
	round[5].diff_lay	15cad8053921e366b22527e263ac1655
6	round[6].start	15cad8053921e366b22527e263ac1655
	round[6].key_add	aa0a3c529739c399cc2bdc5fe377332b
	round[6].s_box	62dbeb8885aa2e02271786534d19c3ae
	round[6].diff_lay	c99046c53a2a4e5d0fe0777d157e4210
7	round[7].start	c99046c53a2a4e5d0fe0777d157e4210
	round[7].key_add	25d3bf894f4658f0bd4ca1360cc887c8
	round[7].s_box	3feff4f884ad5e257af9f164fe35ea4a
	round[7].diff_lay	7e4b14fdd83b2d9cff53fa40030299f3
8	round[8].start	7e4b14fdd83b2d9cff53fa40030299f3
	round[8].key_add	feb32a69c4fe63aacf87fa08d942b124
	round[8].s_box	0cf6e58e88acfb395f612d62e576c855
	round[8].diff_lay	7dfe7a68e5210f2d45e2409637446d10
9	round[9].start	7dfe7a68e5210f2d45e2409637446d10
	round[9].key_add	861aed854840260e47fc524654ec2251
	round[9].s_box	442e5336522d2362a0af480b20d694c2
	round[9].diff_lay	0af1fe0ac7fb646606dbdf4eb80d3d28
10	round[10].start	0af1fe0ac7fb646606dbdf4eb80d3d28
	round[10].key_add	ea0c01a4ea82833c549bccf94844ac80
	round[10].s_box	bb727ca9bbcdec0ffdde4bded4dd91ec
	round[10].diff_lay	91a588a05488541d077338fa399af522
11	round[11].start	91a588a05488541d077338fa399af522
	round[11].key_add	824cef10a95d7239be08d50cd05a5da1
	round[11].s_box	13f9613ed3a41eaaae62b57270b88de9
	round[11].diff_lay	0a3a6aef6ea7c7cd77df1ab94c3ac71d
12	round[12].start	0a3a6aef6ea7c7cd77df1ab94c3ac71d
	round[12].key_add	05cfa5f7b56a6d05113576150c7ee21b
	round[12].s_box	364506f8d20a3cc2e3c8389781a89803
	round[12].diff_lay	0d678661d5042bdcf0cb803f6343d143
13	round[13].start	0d678661d5042bdcf0cb803f6343d143
	round[13].key_add	60ba3d0d03eb052d64ad4356f76fc1b9
	round[13].s_box	d0a38b097bb036404376647568b1dd7d
	round[13].diff_lay	1d5bea5d7d63d073e0358b7a3539ccb9
14	round[14].start	1d5bea5d7d63d073e0358b7a3539ccb9
	round[14].key_add	e35ace4a123ab14d5f7a3007f4232886
	round[14].s_box	4de58bc83938c82f845104ccba893484
	round[14].diff_lay	5177e22f06f18f9eb71a3a8aae6b2d6b
15	round[15].start	5177e22f06f18f9eb71a3a8aae6b2d6b
	round[15].key_add	34a4be36219897617762d7a4b65732f8

Continues on next page...

		Continued from previous page...
Round	State	Value
	round[15].s_box	18a95a64fd3b85e8f50e0d5d4e28a1f7
	round[15].diff_lay	6ecb38128765cf8676bdaaca83c1483a
16	round[16].start	6ecb38128765cf8676bdaaca83c1483a
	round[16].key_add	19e7f69edc259a7b03aba1e52ccdd95f
	round[16].s_box	8e2e4292939db8d9d571325a424f3553
	round[16].output	00112233445566778899aabbccddeeff