ARIA CIPHER

<Cool-Encrypters>

Department of <CSE>
Indian Institute of Technology Bhilai

November 28, 2020

Outline

Introduction

Abstract :
128-bit block cipher ARIA which is an involution substitution and permutation encryption network(SPN).

Description :

- Key sizes : 128, 192, and 256 bits,
- Block size :128-bit long
- 16×16 involutional binary matrix
- Two kinds of S-boxes of substitution layers
- software and hardware implementation
- Differential cryptanalysis and linear cryptanalysis.

Outline

Specification :

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

State : 128 bit array

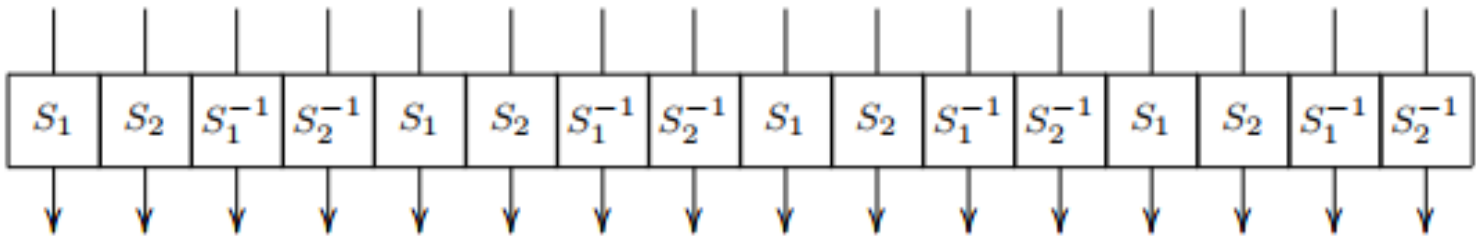
Each round of the cipher consists of the following three parts:

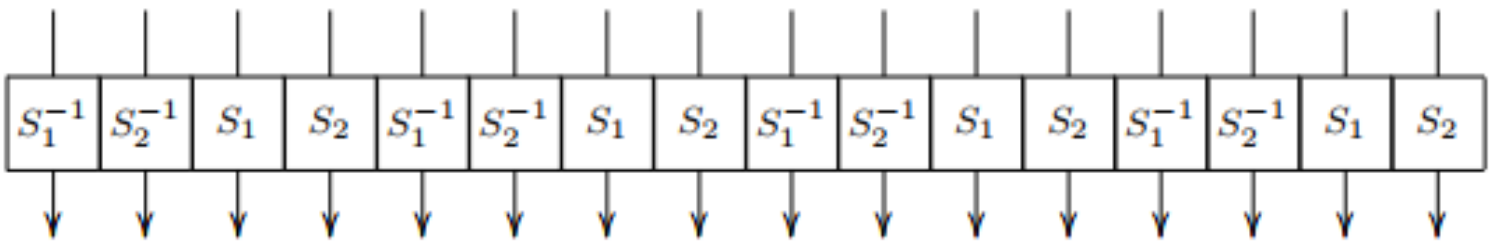
Round key addition :
State is XORed with a 128-bit round key.

Specification :

Substitution layer :

Here 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.





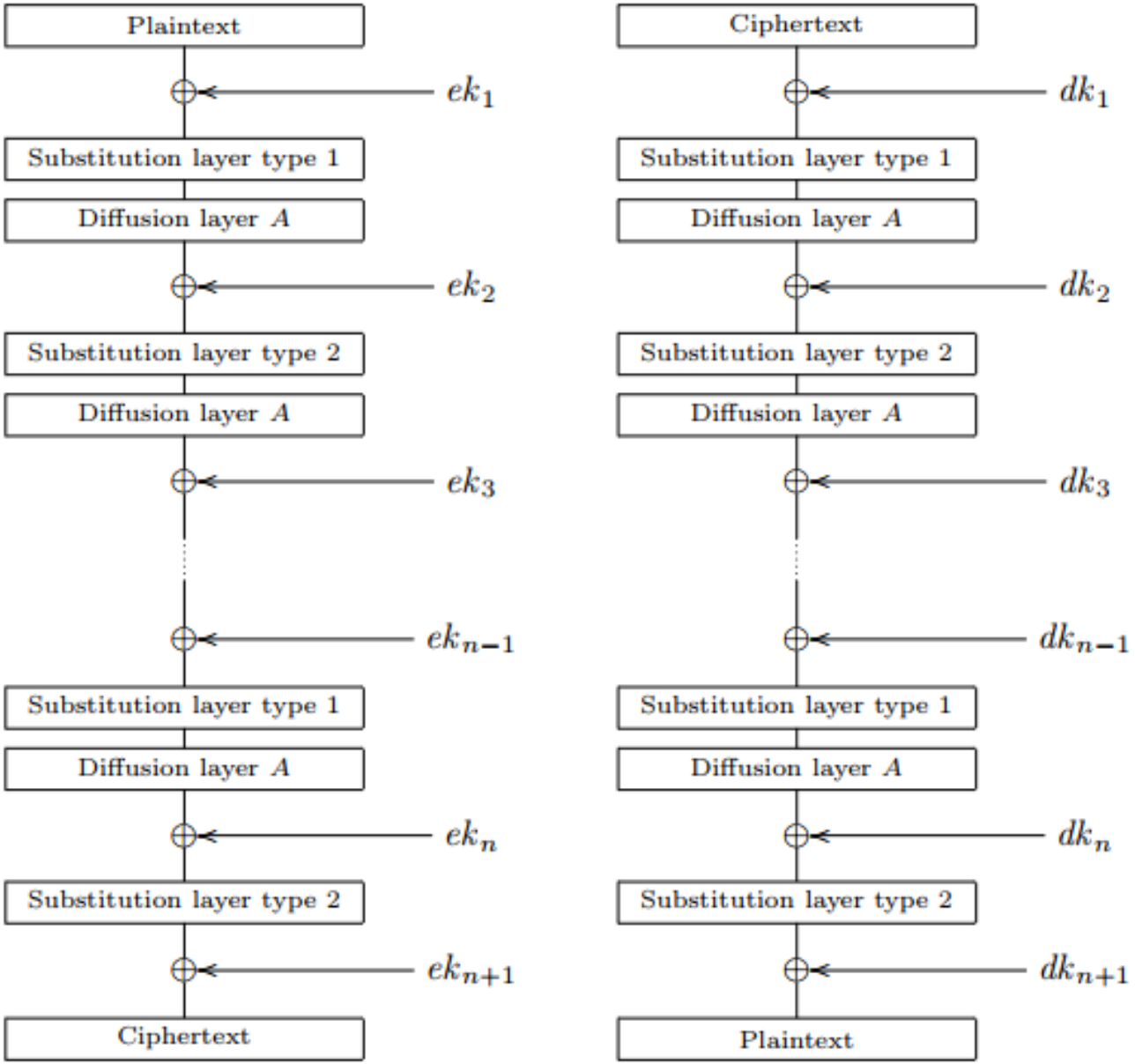
Specification :

Diffusion layer :

A simple 16 $\times$ 16 binary matrix is multiplied to the state, considered as an array of 16 bytes.

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

Specification : The Cipher



Specification :

Key Expansion :

The ARIA key expansion mainly consists of two parts,initialization and round key generation.

1.Initialization
MK is master key
$KL||KR = MK||0..0$

$$W_0 = KL$$
$$W_1 = F_o(W_0, CK_1) \oplus KR$$
$$W_2 = F_e(W_1, CK_3) \oplus W_0$$
$$W_3 = F_o(W_2, CK_3) \oplus W_1$$

Specification :

Key Expansion :

1.Initialization

$$C_1 = 0x517cc1b727220a94fe12abe8fa9a6ee0$$
$$C_2 = 0x6db14acc9e21c820ff28b1d5ef5de2b0$$
$$C_3 = 0xdb92371d2126e970032497750 4e8c90e$$

Then the constants CK_i are defined by the following table:

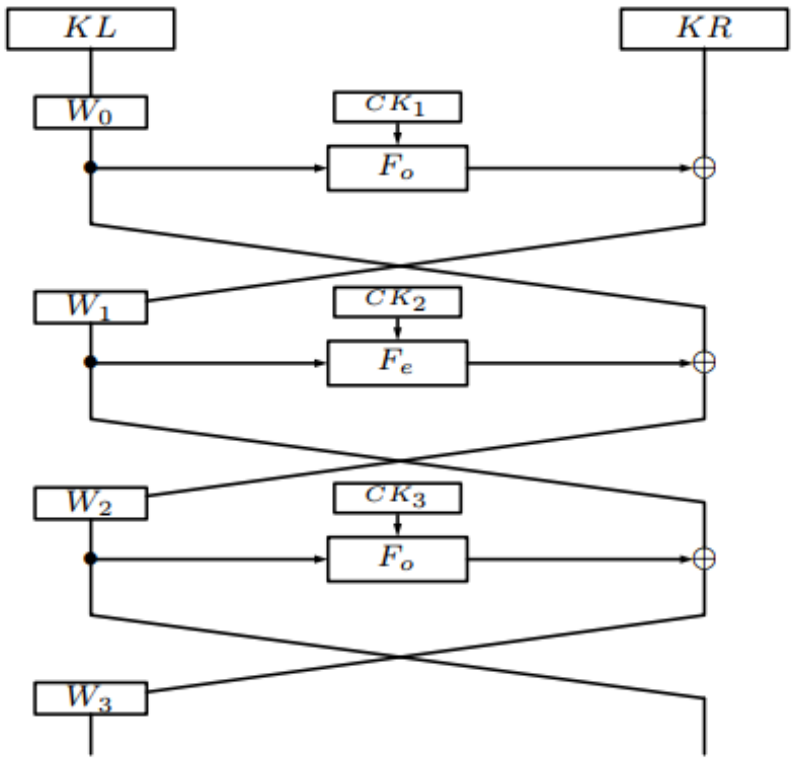
Key Size	CK_1	CK_2	CK_3
128	C_1	C_2	C_3
192	C_2	C_3	C_1
256	C_3	C_1	C_2

Specification :

Key Expansion :

1.Initialization

The complete diagram of initialization process is given below:



Specification :

Key Expansion :

2.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.

$$ek_1 = (W_0) \oplus (W_1 \ggg 19)$$
$$ek_2 = (W_1) \oplus (W_2 \ggg 19)$$
$$ek_3 = (W_2) \oplus (W_3 \ggg 19)$$
$$ek_4 = (W_0 \ggg 19) \oplus (W_3)$$
$$ek_5 = (W_0) \oplus (W_1 \ggg 31)$$

Specification :

Key Expansion :

2.Round Key Generation

Decryption keys generation-

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

Specification :

Software Implementation :

1. 8-bit 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.

Specification :

Software Implementation :

1. 8-bit implementation

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

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

Specification :

Software Implementation :

1. 32-bit implementation

Efficient implementation of diffusion layer only.
Substitution layer already efficient for 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:
$A = M_1 * P * M_1 * M$

Cryptanalysis :

Cryptanalysis is briefly explained in the pdf.The following attacks are mentioned.

1. **Differential and Linear Attack**
2. **Integral Attack**
3. **Multiset/Collision Attacks**
4. **Slide Attacks**

Outline

Observations :

1 – Impossible to calculate the encryption key from partial information of round keys.

2 – The minimum number of active S-boxes with respect to differential and linear cryptanalysis in r-rounds is $8 * \lfloor \frac{r}{2} \rfloor + 2 * (\frac{r}{2} - \lfloor \frac{r}{2} \rfloor)$

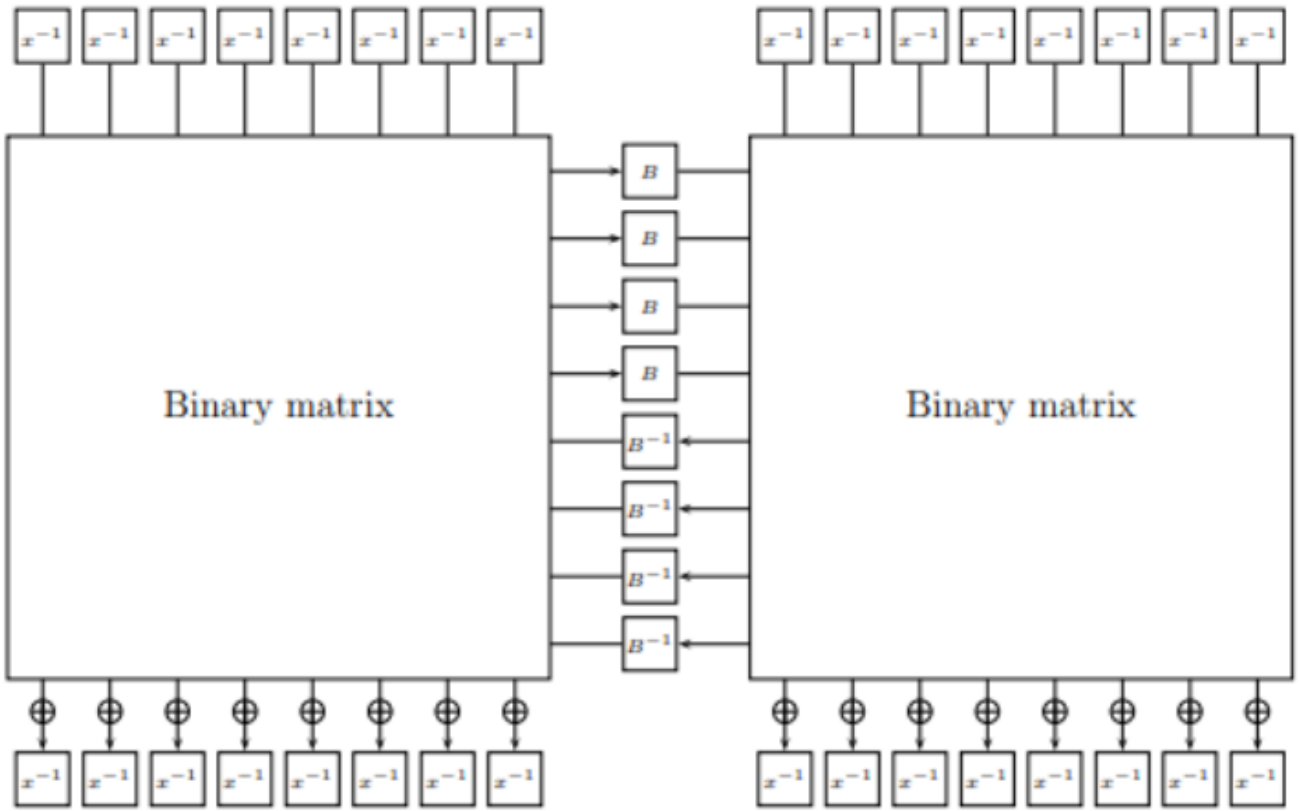
3.Similarity with Rijndael.Inherits strong properties of Rijndael but is also vulnerable to Rijndael attacks.

4.Aria has very specific features (involutional diffusion, special structure of the matrix), thus many new attack methods can be introduce over here.

Outline

Brownie Point Nominations :

Splitting of Cipher



Brownie Point Nominations :

Splitting of Cipher

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).

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).

Outline

Conclusions :

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 an involutional structure like in diffusion layer for efficient implementations on various environments.
- uses fiestel cipher to generate secure round keys.

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

Thanks

Team Members

- Divyansh Khandelwal
- Tanish Gupta
- Driti Singh

Implementation Info

- Github Link:- https://github.com/tanish265/ARIA-Cipher