Trends in Mathematics
Information Center for Mathematical Sciences
Volume 8, Number 1, June, 2005, Pages 13–20

DESIGN OF BLOCK CIPHERS AND CODING THEORY

DAESUNG KWON, SOO HAK SUNG, JUNG HWAN SONG AND SANGWOO PARK

Abstract. In this article, we review the designs of diffusion layers of blocks ciphers based on linear codes. All linear codes are not suitable for diffusion layers. To use as diffusion layers, the linear codes should be best among such codes, from the view point of efficiency and security. The well-known block cipher AES uses a MDS(Maximal Distance Separable)-code and Camellia and ARIA use MDBL(Maximal Distance Binary Linear)-codes.

As an example, we describe the choice rationale of the diffusion layer of ARIA which is a MDBL $[32, 16, 8]$-code in detail.

1. Introduction

A block cipher is a reversible function

$$g : K \times B \to C$$

which maps a key in K and a block in B into a block in C where B, C are set of blocks of fixed-length. Security of a cipher could be described as a difficulty of calculation the inverse of

$$\bar{g} : K \to Map(B, C), \quad \bar{g}(k)(b) = g(k, b).$$

Most block ciphers are constructed by composing several simple functions - and iterated block cipher. Each iteration is called a round and a function is termed a round function.

$$g = \underbrace{f \circ f \circ \cdots \circ f}_{r \text{ times}}.$$

The number of rounds depends on the desired security level and the consequent trade-off with performance.

Many block ciphers can be categorized as Feistel networks, or more generally as SPN(substitution-permutation networks).

An SPN structure is widely used for designing a block cipher such as AES[9] and is used for a round function of a Fiestel cipher such as Camellia[1]. An SPN structure is consisting of a substitution layer followed by a permutation function. A permutation function is also called diffusion layer.

Popular choices of substitution function are affine transforms of x^{-1} on $GF(2^8)$. Such s-boxed are widely used in AES, Camellia, ARIA, etc. N. T. Courtois claims that this construction may weak against algebraic attacks[5]. But, recently, C. Diem showed that the claim is not true[8].

Key words and phrases. Block ciphers, Diffusion layers, Linear codes, MDS-codes, MDBL-cods, AES, ARIA.

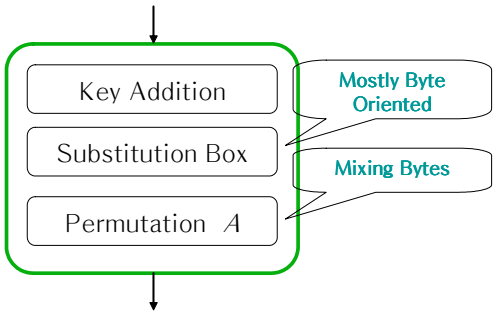

FIGURE 1. Substitution-permutation networks

The constructions of permutation functions are various. Distinguishing functions are those used in AES, Camellia, ARIA.

TABLE 1. Permutation functions of AES, Camellia, ARIA

Block cipher	Permutation function
AES	4×4 matrix over $GF(2^8)$
Camellia	8×8 binary matrix over $GF(2^8)$
ARIA	16×16 binary matrix over $GF(2^8)$

To choose a matrix, one should determine which matrix is proper as permutation function from the viewpoint of efficiency and security. From the viewpoint of security, the branch number of a matrix is important which is representing the diffusion rate and measures security against differential[2] and linear cryptanalysis[17]. From the viewpoint of efficiency, the matrix should be efficient in hardware and software implementations.

In the rest of paper, firstly, we describe the branch numbers of permutation functions and the relations between them and the distances of linear codes. Secondly, we present the design rationale of permutation function of ARIA from the point of efficiency.

2. Branch numbers of matrices and distance of linear codes

The branch number of a permutation function is representing the diffusion rate and measures security against differential[2] and linear cryptanalysis[17]. Most of diffusion layers are linear transformations and represented as matrices. Let n be the number of s-boxes in a diffusion layer A where the size of input and output of each s-box is m-bit. Define a diffusion layer $A : (\{0,1\}^m)^n \to (\{0,1\}^m)^n$ which is a linear transformation as follow:

$$A(x) = A \cdot x^T = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

where $x = (x_1, x_2, \cdots, x_n)^T, x_i \in \{0,1\}^m, i = 1, \cdots, n$. The branch number of A is defined as follows

Definition 1. *The branch number of an $n \times n$ matrix A is defined by*

$$\beta(A) = min\{wt(x) + wt(A \cdot x^T) | x \in (\{0,1\}^m)^n, x \neq 0\}. \tag{1}$$

A linear $[n,k]$-code C is a subspace of $GF(q)^n$ of dimension k and represented by $k \times n$ matrix where rows of are linearly independent, which is called a *generating matrix*. A generating matrix is equivalent to a matrix of the form $[I_{k\times k}, B]$ where $I_{k\times k}$ is a $k \times k$ identity matrix and B is a $k \times (n-k)$ matrix. The form $[I_{k\times k}, B]$ of a matrix is called an *standard form* of a generating matrix. Therefore a linear code is generated by a standard form of a generating matrix.

The Hamming weight of a code word $c \in C$ is the number of nonzero components in c and denoted by $wt(c)$. The distance $d(C)$ of a linear code C is defined by

$$\begin{aligned} d(C) &= \min\{wt(c) | c \in C, c \neq 0\} \\ &= \min\{wt(c', c'B) | c' \in GF(q^m), c' \neq 0\} \\ &= \min\{wt(c') + w(c'B) | c' \in GF(q^m), c' \neq 0\} \end{aligned}$$

A linear $[n,k,d]$-code C represents a linear $[n,k]$-code whose distance is d.

Definition 2. *A linear $[n,k,d]$-code C is said to be a maximal distance separable (MDS) code if*

$$d = n - k + 1.$$

The branch number of a matrix and the distance of linear code are related as follows:

Theorem 3. *Let C be a MDS $[2n,n,d]$-code over $GF(2^8)$ and $[I_{n\times n}, B]$ be a standard form. Then,*

$$d(C) = Br(B^T)$$

Proof. The proof can be derived from definition as follows:

$$\begin{aligned} n + 1 &= d(C) \\ &= \min_{c' \in GF(2^m)^n, c' \neq 0} (w(c') + w(c'B)) \\ &= Br(B^T) \end{aligned}$$

□ □

MDS codes[17] can be used for determining the maximum branch number of matrices in $GF(2^m)^{n\times n} = (\{0,1\}^m)^{n\times n}$ and for finding a matrix with maximum branch number when m is greater than 1. A 4×4 matrix over $GF(2^8)$ is used in AES, and an 8×8 matrix over $GF(2^8)$ is used in Khazad[7].

However, the result is not necessarily true for binary matrices, i.e., matrices in $GF(2)^{n\times n}$. For example, in [10], Kanda et al. proved that the maximum branch number for 8×8 binary matrices is 5 through computer simulations, while the maximum branch number of matrices in $GF(2^m)^{8\times 8} (m > 1)$ is 9. Recently, in [18], Sung et al. present the relations between the branch numbers of binary matrices

and the distances of binary linear $[2n, n]$-codes which are linear $[2n, n]$-codes whose base field is $GF(2)$.

Let $G_{n\times 2n} = [I_{n\times n}, A_{n\times n}]$ be the echelon form of generating matrix of linear $[2n, n, d]$-code over $GF(2^m)$, then the branch number of A^T is d by the definition of branch number.

Theorem 4. [18] *If $G_{n\times 2n} = [I_{n\times n}, A_{n\times n}]$ is the echelon form of generating matrix of binary linear $[2n, n, d]$-code, then the branch number of A^T is d.*

From Theorem 4, we see that the maximum branch number of $n \times n$ binary matrices is equal to the maximum distance of binary linear $[2n, n]$-codes. It is an important topic in the coding theory to find the maximum distance of binary linear $[2n, n]$-codes. The upper and lower bounds for the maximum distance are summarized by Brouwer [3]. Some upper and lower bounds for the maximum distance are stated in Table 2. When $n \leq 18$, we can know the exact maximum distance. From this fact and Theorem 4, we can also know the maximum branch number for $n \times n (n \leq 18)$ binary matrices and the lower bounds and upper bounds for $n > 18$.

TABLE 2. Maximum distance of binary linear $[2n, n]$-codes

n	lower bound	upper bound	n	lower bound	upper bound
1	2	2	17	8	8
2	2	2	18	8	8
3	3	3	19	8	9
4	4	4	20	9	10
5	4	4	21	10	10
6	4	4	22	10	10
7	4	4	23	11	11
8	5	5	24	12	12
9	6	6	25	10	12
10	6	6	26	10	12
11	7	7	27	11	13
12	8	8	28	12	14
13	7	7	29	12	14
14	8	8	30	12	14
15	8	8	31	12	15
16	8	8	32	12	16

Remark 5. *From Theorem 4 and Table 2, we conclude that the maximal branch number of 8×8 binary matrices is 5 and that of 16×16 binary matrices is 8.*

3. Design rationale of the diffusion layer of ARIA

In this section, we describe the choice rationale of the diffusion layer (permutation function) of ARIA[14] from the viewpoint of efficiency. The diffusion layer

$A:\ \mathrm{GF}(2^8)^{16} \to \mathrm{GF}(2^8)^{16}$ of ARIA is given by

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

where

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

An equivalent expression would be given by a matrix multiplication as follow:

(2)

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

Since we want the cipher to be involutional and all states to be mixed by the diffusion layer, in [13, 14], they chose a matrix which satisfies the following properties.

- It should be involutional.
- The branch number should be 8 (maximal).
- It should be suitable for implementations on 8-bit/32-bit processors.
- It should be suitable for hardware implementations.

We can easily show that the branch number of the matrix in (2) is 8. To show the efficiency in various platforms, we present some techniques[12] which optimize implementations on 8-bit processors such as Smart Cards and on 32-bit processors such as PCs.

3.1. **8-bit based implementations.** On an 8-bit processor, except for S-box substitution, all operations are XOR's. The implementation of S-box substitution requires four tables of 256 bytes.

In a straight coding, 112 XOR's are required to implement one round except for S-box substitutions. We present a method to reduce the number of operations to 76 XOR's using four additional variables $T_1, \cdots, T_4$ as follows.

$$
\begin{aligned}
&T_1 = x_4 \oplus x_5 \oplus x_{10} \oplus x_{15}, \quad T_2 = x_3 \oplus x_6 \oplus x_9 \oplus x_{16},\\
&y_1 = x_7 \oplus x_9 \oplus x_{14} \oplus T_1, \quad y_2 = x_8 \oplus x_{10} \oplus x_{13} \oplus T_2,\\
&y_6 = x_2 \oplus x_{11} \oplus x_{16} \oplus T_1, \quad y_5 = x_1 \oplus x_{12} \oplus x_{15} \oplus T_2,\\
&y_{12} = x_3 \oplus x_8 \oplus x_{13} \oplus T_1, \quad y_{11} = x_4 \oplus x_7 \oplus x_{14} \oplus T_2,\\
&y_{15} = x_1 \oplus x_6 \oplus x_{12} \oplus T_1, \quad y_{16} = x_2 \oplus x_5 \oplus x_{11} \oplus T_2,\\
\\
&T_3 = x_2 \oplus x_7 \oplus x_{12} \oplus x_{13}, \quad T_4 = x_1 \oplus x_8 \oplus x_{11} \oplus x_{14},\\
&y_3 = x_5 \oplus x_{11} \oplus x_{16} \oplus T_3, \quad y_4 = x_6 \oplus x_{12} \oplus x_{15} \oplus T_4,\\
&y_8 = x_4 \oplus x_9 \oplus x_{14} \oplus T_3, \quad y_7 = x_3 \oplus x_{10} \oplus x_{13} \oplus T_4,\\
&y_{10} = x_1 \oplus x_6 \oplus x_{15} \oplus T_3, \quad y_9 = x_2 \oplus x_5 \oplus x_{16} \oplus T_4,\\
&y_{13} = x_3 \oplus x_8 \oplus x_{10} \oplus T_3, \quad y_{14} = x_4 \oplus x_{70} \oplus x_9 \oplus T_4.
\end{aligned}
$$

If it is implemented serially, only one extra variable is required in the method.

3.2. **Software implementations on 32-bit processors.** On 32-bit processors, Rijndael and Camellia can be implemented efficiently by combining S-box substitutions and the diffusion layer by 8×32 table lookups. This technique is suitable for a diffusion layer which is based on 32-bits. Since the diffusion layer of ARIA is based on 128-bits, on 32-bit processors, it looks less efficient than Rijndael and Camellia. However, we choose a matrix in 16×16 binary matrices which can be efficiently implemented on 32-bit processors.

Since M_1 is an involution($M_1^{-1} = M_1$ in Section 3.2) The diffusion layer A is chosen in the form of $M_1 \cdot M_2 \cdot M_1$ to have an involution structure where

$$
M_1 = \begin{pmatrix} I & I & I & 0 \\ I & 0 & I & I \\ I & I & 0 & I \\ 0 & I & I & I \end{pmatrix}, \quad M_2 = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 \\ 0 & 0 & 0 & P_3 \end{pmatrix} \cdot \begin{pmatrix} T & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{pmatrix},
$$

for the 4×4 identity matrix I and following four 4×4 matrices

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

For simplifying the above notations, we use the following notations

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

If we let S be the S-box substitution, the one round except for key addition can be written as $A \cdot S = M_1 \cdot M_2 \cdot M_1 \cdot S$. It is easy to see that the form of $M_1 \cdot S$ is not implemented efficiently by 8×32 table lookups. So, we make the following modification for the representation of the diffusion layer as follows:

$$
\begin{aligned}
A &= M_1 \cdot M_2 \cdot M_1 = M_1 \cdot P \cdot M \cdot M_1 = M_1 \cdot P \cdot M \cdot M_1 \cdot M \cdot M \\
&= M_1 \cdot P \cdot M \cdot M \cdot M_1 \cdot M = M_1 \cdot P \cdot M_1 \cdot M.
\end{aligned}
$$

Since M is a block diagonal matrix, $M \cdot S$ can be implemented by 8×32 table lookups. It is clear that M_1 is implemented by simple 32-bit word operations. The operation P is a byte-oriented operation that is done within each 32-bit word. Hence this gives a way to implement ARIA on a 32-bit based machine.

3.3. **Hardware implementations.** In hardware implementations, operations can be proceeded in parallel. Therefore, the performance is determined by the critical paths. While the critical path of the AES permutation function is 4 2-input XOR gates, that of the ARIA is 3 2-input XOR gates. Therefore, ARIA permutation function is more efficient than that of AES.

4. Conclusion

The linear codes are used in the construction of permutation functions of block ciphers. In particular, MDS-codes, MDBL-codes are used in AES, Khazad, Camellia, Although the choice of good codes is difficult, other codes could be used in the design of good block ciphers.

References

[1] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, Camellia: A 128-bit block cipher suitable for multiple platforms, available at http://info.isl.ntt.co.jp/camellia/.

[2] E. Biham and A. Shamir, Differential cryptanalysis of DES-like cryptosystems, Advances in Cryptology - Crypto'90, (Springer-Verlag, 1991), pp. 2-21.

[3] A.E. Brouwer, Bounds on the maximum distance of linear codes, available at http://www.win.tue.nl/ ∼aeb/voorlincod.html/.

[4] K. Chun, S. Kim, S. Lee, S.H. Sung, and S. Yoon, Differential and linear cryptanalysis for 2-round SPNs, Inform. Process. Lett. 87 (2003), pp 277-282.

[5] N. T. Courtois, and J. Pierprzy, Cryptanalysis of Block Ciphers with Overdefined Systems of Equations, Advances in Cryptology - Asiacrypt 2002, (Springer-Verlag, 2003), pp. 267-287.

[6] J. Daemen, R. Govaerts, and J. Vandewlle, Correlation matrices, Fast Software Encryption, LNCS 1008, (Springer-Verlag, 1994), pp. 275-285.

[7] P. S. L. M. Barreto and V. Rijmen, *T*he Khazad legacy-level block cipher. Primitive submitted to NESSIE, Sept. 2000.

[8] C. Diem, The XL-Algorithm and a Conjecture from Commutative Algebra, Accepted by Asiacrypt 2002, Dec 2004.

[9] FIPS 197, The Advanced Encryption Standard (AES), Federal Information Processing Standard, NIST, US Department of Commerce, Washington DC, 2001.

[10] M. Kanda, Y. Takashima, T. Matsumoto, K. Aoki, and K. Ohta, A strategy for construction fast round functions with practical security against differential and linear cryptanalysis, Selected Areas in Cryptography-SAC'98, LNCS 1556, (Springer-Verlag, 1999), pp. 264-279.

[11] J.-S. Kang, S. Hong, S. Lee, O. Yi, C. Park, and J. Lim, Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks, ETRI J. 23 (2001) pp. 158-167.

[12] BonWook Koo, HwanSeok Jang, JungHwan Song. Constructing and Cryptanalysis of a 16x16 Binary Matrix as a Diffusion Layer. , editors, *WISA2003*, *Lecture Notes in Computer Science*, Springer, 2003.

[13] D. Kwon, J. Kim, S. Park, S.H. Sung, Y. Sohn, J.H. Song, Y. Yeom, E-J. Yoon, S, Lee, J. Lee, S. Chee, D. Han, and J. Hong, New block cipher: ARIA, ICISC 2003, LNCS 2971, pp. 432-445, 2004.

[14] Specification of ARIA, available at http://www.nsri.re.kr/ARIA.

[15] M. Matsui, Linear cryptanalysis method for DES cipher, Advances in Cryptology - Eurocrypt'93, (Springer-Verlag, 1994), pp. 386-397.
[16] NTT-Nippon Telegraph and Telephone Corporation, Specification of E2- a 128-bit block cipher, 1999, available at http://info.isl.ntt.co.jp/e2/.
[17] V. Rijmen, J. Daemen, B. Preneel, and A. Bosselaers, The cipher Shark, Fast Software Encryption-FSE'96, LNCS 1039, (Springer-Verlag, 1996), pp. 99-112.
[18] S. H. Sung, J. H. Song, D. Kwon, Y. Yeom and S. Park, Branch numbers of binary diffusion layers, preprint.

National Security Research Institute,, 161 Gajeong-dong, Yuseong-gu, Daejeon 305-350, KOREA, Department of Computing information & mathematics, Paichai University,, 426-6 Doma-dong, Seo-gu, Daejeon 302-735 KOREA, Department of Mathematics, Hanyang University,, 17 Haengdang-dong, Seongdong-gu, Seoul 133-791, KOREA

E-mail address: ds_kwon, psw@etri.re.kr

E-mail address: sungsh@mail.pcu.ac.kr

E-mail address: camp123@hanyang.ac.kr