Feistel Like Construction of Involutory Binary Matrices With High Branch Number

Adnan Baysal[1,2], Mustafa Çoban[3], and Mehmet Özen[3]

[1]TÜBİTAK - BİLGEM, PK 74, 41470, Gebze, Kocaeli, Turkey, adnan.baysal@tubitak.gov.tr
[2]Kocaeli University, Department of Computer Engineering, Faculty of Engineering, Institute of Science, 41380, Umuttepe, Kocaeli, Turkey
[3]Sakarya University, Faculty of Arts and Sciences, Department of Mathematics, Sakarya, Turkey, cobanmus@yahoo.com, ozen@sakarya.edu.tr

August 4, 2016

Abstract

In this paper, we propose a generic method to construct involutory binary matrices from a three round Feistel scheme with a linear round function. We prove bounds on the maximum achievable branch number (BN) and the number of fixed points of our construction. We also define two families of efficiently implementable round functions to be used in our method. The usage of these families in the proposed method produces matrices achieving the proven bounds on branch numbers and fixed points. Moreover, we show that BN of the transpose matrix is the same with the original matrix for the function families we defined. Some of the generated matrices are *Maximum Distance Binary Linear* (MDBL), i.e. matrices with the highest achievable BN. The number of fixed points of the generated matrices are close to the expected value for a random involution. Generated matrices are especially suitable for utilising in bitslice block ciphers and hash functions. They can be implemented efficiently in many platforms, from low cost CPUs to dedicated hardware.

Keywords: Diffusion layer, bitslice cipher, hash function, involution, MDBL matrices, Fixed points.

1 Introduction

Modern block ciphers and hash functions use two basic layers iteratively to provide security: *confusion* and *diffusion*. Confusion is done by non-linear sub-

stitutions implemented as look-up-tables or Boolean functions of (usually) small input size. These functions are called as a substitution box (abbreviated as S-box). There are many S-box generation methods proposed in the literature such as inversion in $GF(2^n)$ [11], power functions in $GF(2^n)$ [26], modular multiplication with a key [22] and using small S-boxes in a mini cipher to generate a greater S-box [8].

On the other hand, diffusion layers aim to mix the bits not diffused in confusion layer. Commonly a linear function is applied in this layer. Confusion and diffusion layers should be compatible with each other to provide security against known cryptanalysis methods in as fewer rounds as possible. It is better if the security can be proved by means of a widely accepted metric such as minimum active S-box count in differential and linear trails [19]. For block cipher design, inverses of the confusion and diffusion layer may be necessary for decryption. In such a case, both confusion and diffusion layers and their inverses have to be efficient.

1.1 Related work

Each linear diffusion layer can be represented by a matrix multiplication. Properties of the matrix directly affect the security of the block cipher. For example, *Advanced Encryption Standard* (AES) [11] uses a 4×4 *Maximum Distance Separable* (MDS) matrix in $GF(2^8)$ together with a byte permutation (shift rows) layer. This combination ensures a high number of differentially and linearly active S-boxes over consecutive rounds.

The success of AES encouraged researchers to generate efficient MDS codes for different implementation mediums. For low area hardware designs, companion matrices are proposed [14] and analysed [17] where multiple round application of a linear feedback shift register gives an MDS matrix. Some MDS matrices which have simple inverses are proposed in [28] by the use of simple linear functions and a Feistel like structure. A similar method was applied in [33], where an LFSR with matrix coefficients in the polynomial ring over binary matrices is used. In [27, 16, 30], involutory MDS matrices are generated, which reduces decryption overhead of the inverse diffusion layer.

Another direction in diffusion layer construction is creating *Maximum Distance Binary Linear* (MDBL) codes. ARIA block cipher [21] is one such example, where an 16×16 involutory MDBL matrix is used over bytes, i.e., over $GF(2^8)$. In the diffusion layer of ARIA, only XOR operation on bytes is used which reduces implementation cost when compared to field multiplication as used in AES' MDS matrix. Again this construction ensures a high number of active S-boxes in 2 consecutive rounds of an Substitution-Permutation-Network-like (SPN-like) cipher.

The expected number of fixed points for a random involution is about $2^{\frac{n}{2}}$ [7]. These fix points may result in iterative differential and linear characteristics depending on the S-box layer. Therefore, reducing fixed points is another problem in diffusion layer design. There are 2^{72} fixed points in the diffusion layer of ARIA [34], which is 2^8 times higher than the expected value of 2^{64}. There are

some work on designing binary matrices as in ARIA for different lengths and with the aim of reducing the number of fixed points [20, 2, 29].

ARIA-like diffusion layers are suitable for *Substitution Permutation Network* (SPN) type ciphers where for some positive integer σ, consecutive σ-bit blocks enter into S-boxes and diffusion layer mixes the words which are outputs of S-boxes. There is another type of SPN in which S-box layer is defined as bit wise Boolean operations on t words of bit size w. This type of S-box layer is called a *bit-slice* S-box layer, and a cipher or hash function using this S-box is called a bit-slice cipher or hash function. In this approach, i^{th} bit of each word is incorporated into an S-box, hence w S-box applications can be done in parallel in a CPU with w-bit words. Hence, S-box bits are not consecutive but interleaved by w-bits in this type of ciphers and hash functions. If t is small (*e.g.*, three or four), this S-box layer can be implemented efficiently both in hardware and software. Since bit positions are manipulated by the confusion layer, diffusion layer must diffuse bits inside the words. Therefore, ARIA-type word-wise XOR matrices may be inefficient for bitslice ciphers if not implemented as look-up tables.

Block ciphers with bitslice S-box layers such as LS-design ciphers [13], PRIDE [1], and RoadRunneR [4] use binary matrices that operate inside the word. In SHA-3 competition [31], the winner algorithm Keccak [5] and one of the finalists JH [32] also use bitslice S-box layers, but diffusion inside the words is done by word-wise permutations in these algorithms. In [1] and [4], heuristic and exhaustive methods are applied to find diffusion layers with low implementation cost and relatively high branch number (BN). On the other hand, LS-design [13] offers the use of look-up tables for diffusion layer, which may be costly in some implementation environments such as low cost RFIDs. Hence it is important to find generic and efficient methods of creating binary matrices for such designs, but this subject is understudied in the literature. Recently, in [15], Guo *et. al.* analysed the construction of diffusion layers based on Feistel structure with different permutations in each round. Our construction, on the other hand, fixes the round number to three, and makes use of efficient linear mappings as Feistel round functions. We also used the same function in each round.

1.2 Contributions

We propose a new iterative method for generating area efficient involutory binary matrices from a three-round Feistel scheme. We prove some bounds on branch number and number of fixed points for the proposed method. These bounds are achieved by two families of efficiently implementable round functions we defined. We also prove that the defined families generate diffusion layers with the same BN for the transpose matrix. Some of the generated matrices satisfy the MDBL property, i.e., have the maximum branch number possible for a binary matrix. For some bit sizes that the maximum BN is not known, matrices with known maximal BN values are found. We also discuss the efficiency of the proposed method in hardware and software.

The rest of the paper is organised as follows: the iterative method we pro-

pose for generating involutory binary matrices with high branch number using smaller matrices is presented in Section 2. Theoretical bounds on the BN and number of fixed points are given in Section 3. In Section 4, two sets of efficiently implementable F functions used in our experiments are defined, and experimental results of BN values using this two family is summarised in Section 5. The efficiency of our matrices for different implementation platforms is discussed in Section 6. Section 7 concludes the paper and proposes some future work.

2 Feistel-like Binary Diffusion Layer Construction

Before explaining the proposed method, we discuss some security issues related to diffusion layers in the following section.

2.1 Security related properties of binary linear diffusion layers

One of the most important security metrics of a binary diffusion layer matrix for bitslice and ARIA like block ciphers and hash functions is BN [10], which is formally defined below:

Definition 1 (Branch Number)**.** *The Branch Number (BN) of an $n \times n$ binary matrix $\mathbf{M}$ where $n > 0$ is defined as follows:*

$$BN(\mathbf{M}) = \min_{\mathbf{a} \in GF(2^n)^*} \{|\mathbf{a}| + |\mathbf{M} \cdot \mathbf{a}|\}$$

where $|\mathbf{a}|$ and $|\mathbf{M} \cdot \mathbf{a}|$ is the Hamming weight (number of ones) of the column vector $\mathbf{a}$ and $|\mathbf{M} \cdot \mathbf{a}|$ respectively, $\cdot$ is the matrix multiplication operation (matrix on the left, column vector on the right), and $GF(2^n)^*$ is the set of non-zero vectors in $GF(2^n)$.

BN determines the minimum number of differentially active S-boxes in two consecutive rounds. This minimum number can be used to find a lower bound on the minimum number of active S-boxes for more rounds. Similarly, for linearly active S-box count, the BN of the transpose matrix should be calculated. These two numbers need not to be the same for an arbitrary matrix, but they are the same for our family of round functions. Therefore, whenever we mention BN, we mean both the BN of the original and transpose matrices. These minimum active S-box counts should be maximised to provide better security in terms of differential attack [6] and linear attack [24], which are two basic attacks on block ciphers.

The data complexity of differential attack and linear attack increases if more S-boxes are active. Using calculated bounds on minimum number of active S-boxes in multiple rounds, it is possible to determine a lower bound on round number selection of the cipher. Increasing the number of rounds increases the

number of active S-boxes, but this also reduces the performance. Hence, a diffusion layer with good performance and high BN is necessary to build an efficient and lightweight block cipher or hash function. Note that if the final swap is not omitted in three round Feistel scheme, the BN of the big matrix is not affected.

Another metric of the security of a diffusion layer is the set of fixed points of the diffusion layer, i.e. the inputs $\mathbf{a}$ satisfying $\mathbf{M} \cdot \mathbf{a} = \mathbf{a}$ for the diffusion layer $\mathbf{M}$. There is an argument stating that fixed points are points with no diffusion [34], but we believe that diffusion should be understood as the effect of input bits to the result of output bits in Boolean terms. Each input bit diffuses to output bits that it affects in the corresponding matrix multiplication. Hence, in our point of view, diffusion is unrelated with fixed points. Whereas, fixed points may be used to generate iterative differential and linear characteristics. Hence the number of fixed points and their value may be efficient on the security of the block cipher or hash function using the matrix in its diffusion layer.

2.2 Proposed method

The area overhead of the inverse of a diffusion layer becomes vital when both the encryption and the decryption algorithms of a block cipher are needed to be implemented on a resource restricted device. In block cipher design, decryption overhead can be minimised by using Feistel schemes. We applied the same idea in diffusion layer construction.

In classical Feistel scheme, input is divided into two halves (left and right). At each round, a copy of one of the halves enters into a round function F, and the result is XORed to the other part. Output of XOR and the original input to F is swapped after this operation. These operations are applied multiple rounds and in the last round, swap is omitted. As a result, the inverse algorithm is the same. A three round Feistel scheme is visualised in Figure 1.

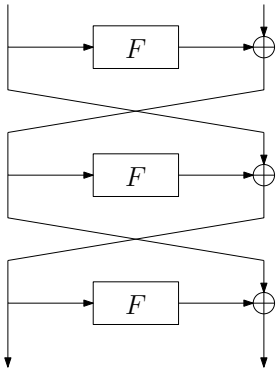

Figure 1: Classical three round Feistel scheme.

We propose to use linear F function in this three-round Feistel scheme to

generate a $2n$-bit diffusion layer from n-bit linear functions. This approach have two important benefits:

1. Inverse function have the same algorithm or circuit
2. Area cost may be reduced since only half of the data is processed

The disadvantage is the fact that three rounds require more clock cycles than a single round matrix. A similar idea to our generation method was used in [8] where $2n$-bit S-box is generated using n-bit S-boxes in three rounds Feistel and MISTY [25] type structures. We did not analyse MISTY-type structure since this construction requires inverse of the small matrix. Because of the fact that area complexity of a random S-box grows exponentially in bit size n, generated S-boxes' bit size cannot be increased too much. In their paper [8], authors showed examples of 8-bit S-boxes constructed from 4-bit S-boxes.

In our matrix generation method, we aim to increase the BN using matrices of smaller BN. The increase in the BN from small matrix (Feistel F function) to big matrix (three round Feistel) is called as *BN gain* in the paper.

Design rationale of round number selection : We select the round number to be three since two rounds is not enough to gain a significant BN advantage in our family of linear round functions defined in Section 4. The following theorem classifies the linear F functions which has small BN gain when used in two rounds of Feistel scheme:

Theorem 1. *Let* $\mathbf{M}$ *be an* $n \times n$ *binary matrix where* n *is a positive integer and* $BN(\mathbf{M}) = d$*. Assume that there exists a vector* $\mathbf{a} \in GF(2^n)$ *such that* $|\mathbf{a}| = 1$ *and* $|\mathbf{a}| + |\mathbf{M} \cdot \mathbf{a}| = d$*. If* $\mathbf{M}$ *is used as* F *function in two rounds of Feistel scheme and resultant* $(2n) \times (2n)$ *matrix is called as* $\mathbf{D_M}$*, then* $BN(\mathbf{D_M}) \leq d + 1$*.*

Proof. Let $\mathbf{a}$ be a vector as defined in the theorem. Then $\mathbf{D_M} \cdot \begin{bmatrix} 0 \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \mathbf{M} \cdot \mathbf{a} \end{bmatrix}$. Since $|\mathbf{a}| = 1$ and $|\mathbf{a}| + |\mathbf{M} \cdot \mathbf{a}| = d$, $BN(\mathbf{D_M}) \leq d + 1$. □

By Theorem 1, if there exist vectors with Hamming weight one satisfying the BN of the small matrix, then two rounds of Feistel may increase the BN by at most one. Since we aim high BN gain, two rounds is not sufficient for our case.

On the other hand, a binary matrix $\mathbf{M}$ with $BN(\mathbf{M}) = d$ has at least $d - 1$ ones in each column. Reducing the number of ones in a binary matrix usually decreases its implementation complexity. In this point of view, most efficient matrices has exactly $d - 1$ ones in each column. Therefore, for these efficient matrices, two rounds Feistel may increase the BN by at most one because for all vectors $\mathbf{a}$ with $|\mathbf{a}| = 1$, $|\mathbf{a}| + |\mathbf{M} \cdot \mathbf{a}| = d$.

Moreover, it is a common method to use Feistel scheme in at least three rounds [25, 8]. One reason behind this choice is that at least two F functions are active for an active differential or linear trail, if F is invertible. Hence, the number of rounds is chosen to be three. We did not analyse four round Feistel case because its implementation complexity is higher.

3 Theoretical Bounds on Branch Number and Number of Fixed Points

3.1 On Branch Number

Maximal BN of any $n \times n$ binary square matrix is known only for some small n. For greater n, there are lower and upper bounds. Table 1 shows known lower bounds and proven upper bounds of maximal BN for various n.

Bit size (n)	8	16	32	64
Lower bound	5	8	12	22
Upper bound	5	8	16	28

Table 1: Lower and upper bounds for maximal BN values of binary square matrices depending on bitsize [12].

Assume that $\mathbf{M}$ is an $n \times n$ binary matrix. If $\mathbf{M}$ is used as F function in Figure 1 to generate a $2n \times 2n$ matrix, then the resulting linear function's (call it $\mathbf{D_M}$) matrix representation will be as follows :

$$\mathbf{D_M} = \begin{bmatrix} \mathbf{M}^2 \oplus \mathbf{I}_n & \mathbf{M} \\ \mathbf{M}^3 & \mathbf{M}^2 \oplus \mathbf{I}_n \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{M} \\ \mathbf{B} & \mathbf{A} \end{bmatrix} \tag{1}$$

where $\mathbf{I}_n$ is the identity matrix on n-bit vectors. For the sake of simplicity, we renamed $\mathbf{A} = \mathbf{M}^2 \oplus \mathbf{I}_n$ and $\mathbf{B} = \mathbf{M}^3$. Let $\mathbf{X} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$ be a column vector input to $\mathbf{D_M}$ where $\mathbf{a}, \mathbf{b}$ are n-bit vectors. $\mathbf{a}$ is the most significant word and upper bits denote the most significant bits of each word in our notation. Then, we have the following equation:

$$\mathbf{D_M} \cdot \mathbf{X} = \begin{bmatrix} \mathbf{A} & \mathbf{M} \\ \mathbf{B} & \mathbf{A} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{A} \cdot \mathbf{a} \oplus \mathbf{M} \cdot \mathbf{b} \\ \mathbf{B} \cdot \mathbf{a} \oplus \mathbf{A} \cdot \mathbf{b} \end{bmatrix} \tag{2}$$

For a bit vector or set X, its Hamming weight or cardinality (depending on the context) is denoted by $|X|$. The following theorem finds an upper bound on the binary branch number of $\mathbf{D_M}$:

Theorem 2. $BN(\mathbf{D_M}) \leq 2BN(\mathbf{M})$.

Proof. It is easy to see from Eqn. 2 that for any $\mathbf{a}$, $\begin{bmatrix} \mathbf{a} \\ \mathbf{M} \cdot \mathbf{a} \end{bmatrix}$ is a fixed point of $\mathbf{D_M}$. Let $\mathbf{a}'$ be an n-bit vector such that $|\mathbf{a}'| + |\mathbf{M} \cdot \mathbf{a}'| = BN(\mathbf{M})$. Since $\begin{bmatrix} \mathbf{a}' \\ \mathbf{M} \cdot \mathbf{a}' \end{bmatrix}$ is a fixed point of $\mathbf{D_M}$, we have $BN(\mathbf{D_M}) \leq 2(|\mathbf{a}'| + |\mathbf{M} \cdot \mathbf{a}'|) = 2BN(\mathbf{M})$. □

By Theorem 2, the best achievable BN for generated diffusion layer matrix is double of the BN of the small matrix $\mathbf{M}$. In experiments, we found some $\mathbf{M}$ matrices satisfying this upper bound.

Some criteria on the selection of M matrices : First of all, $\mathbf{M}$ must not be an involution. In such a case $\mathbf{A} = \mathbf{M}^2 \oplus \mathbf{I}_n = \mathbf{I}_n \oplus \mathbf{I}_n = \mathbf{0}_n$, where $\mathbf{0}_n$ denotes $n \times n$ zero matrix, and $\mathbf{B} = \mathbf{M}^3 = \mathbf{M}^2 \cdot \mathbf{M} = \mathbf{I}_n \cdot \mathbf{M} = \mathbf{M}$, so:

$$\mathbf{D_M} = \begin{bmatrix} \mathbf{0}_n & \mathbf{M} \\ \mathbf{M} & \mathbf{0}_n \end{bmatrix}$$

Using this matrix, one gets no advantage on BN, since $BN(\mathbf{D_M}) \leq d = BN(\mathbf{M})$. When $\mathbf{M}^2$ is the zero matrix, so is $\mathbf{M}^3$. Hence resultant $\mathbf{D_M}$ is:

$$\mathbf{D_M} = \begin{bmatrix} \mathbf{I}_n & \mathbf{M} \\ \mathbf{0}_n & \mathbf{I}_n \end{bmatrix}$$

This matrix satisfies $\mathbf{D_M} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \mathbf{0} \end{bmatrix}$ for any $\mathbf{a}$. Therefore, considering vectors $\mathbf{a}$ such that $|\mathbf{a}| = 1$, we can say that $BN(\mathbf{D_M}) \leq 2$.

For $\mathbf{M}^2 = \mathbf{M}$, $\mathbf{D_M}$ satisfies $\mathbf{D_M} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \end{bmatrix}$ for any $\mathbf{a}$. Considering vectors $|\mathbf{a}| = 1$, we get $BN(\mathbf{D_M}) \leq 4$. To sum up, we require $\mathbf{M}^2 \notin \{\mathbf{0}_n, \mathbf{I}_n, \mathbf{M}\}$. Similar arguments can be done when $\mathbf{M}^3 \in \{\mathbf{0}_n, \mathbf{I}_n, \mathbf{M}\}$, this time with different bounds. Thus, these six cases should be avoided.

3.2 Number of Fixed Points

The following theorem classifies and gives the number of fixed points in our method:

Theorem 3. *Let* $\mathbf{M}$ *be an* $n \times n$ *binary matrix,* $ker(\mathbf{M})$ *be the kernel of* $\mathbf{M}$*, and* $\mathbf{D_M}$ *be defined as in Eqn. 1. Then, the number of fixed points of* $\mathbf{D_M}$ *is* $2^n|ker(\mathbf{M})|$.

Proof. Let $\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$ be a fixed point of $\mathbf{D_M}$. Input to the second F function in Figure 1 will be $(\mathbf{M} \cdot \mathbf{a}) \oplus \mathbf{b}$. This input should produce zero vector to make the left hand side $\mathbf{a}$ in the output. Hence $(\mathbf{M} \cdot \mathbf{a}) \oplus \mathbf{b} \in ker(\mathbf{M})$. For any n-bit $\mathbf{a}$ and for any $\alpha \in ker(\mathbf{M})$, there is a corresponding $\mathbf{b} = (\mathbf{M} \cdot \mathbf{a}) \oplus \alpha$. This choice also ensures that right hand side is $\mathbf{b}$ at the output. Hence, these are all the fixed points of $\mathbf{D_M}$. □

Theorem 3 states that there are exactly $2^n|ker(\mathbf{M})|$ fixed points of $\mathbf{D_M}$. Since finding a base of kernel of a matrix has polynomial complexity on bit size n, the number and the set of fixed points can be calculated efficiently. If a differential or linear characteristic is followed using fixed points of the diffusion layer, then there are $2(|\mathbf{a}|+|\mathbf{M}\cdot\mathbf{a}\oplus\alpha|)$ active S-boxes in two consecutive rounds, where $\mathbf{a}$ is any n-bit vector and $\alpha \in ker(\mathbf{M})$. Moreover, these characteristics are iterative. Therefore, if $2(|\mathbf{a}| + |\mathbf{M} \cdot \mathbf{a} \oplus \alpha|) > 2d$, where $BN(\mathbf{D_M}) = 2d$, these iterative characteristics would not give trails with theoretical least possible number of active S-boxes.

Extra swap in the final round : In the last round of three-round Feistel structure, swap is omitted to make the matrix involution. If the last swap is not omitted to simplify implementation (i.e., to use the same function for three rounds), BN is not affected. In the inverse matrix on the other hand, input and output should be swapped. Since inverse of the diffusion layer is not necessary in a hash function design, there would be no extra cost. An important effect of extra swapping in the last round of three-round Feistel is that it changes and probably reduces the number of fixed points. If $\mathbf{X} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$ is a fixed point of this modified version, then the following equations should be satisfied:

$$(\mathbf{M}^3 \oplus \mathbf{I}_n) \cdot \mathbf{a} = (\mathbf{M}^2 \oplus \mathbf{I}_n) \cdot \mathbf{b} \tag{3}$$
$$(\mathbf{M}^2 \oplus \mathbf{I}_n) \cdot \mathbf{a} = (\mathbf{M} \oplus \mathbf{I}_n) \cdot \mathbf{b} \tag{4}$$

If we multiply Eqn. 4 from left by $\mathbf{M}$ and XOR to Eqn. 3 we get:

$$(\mathbf{M} \oplus \mathbf{I}_n) \cdot \mathbf{a} = (\mathbf{M} \oplus \mathbf{I}_n) \cdot \mathbf{b} \tag{5}$$

Combining equations 4 and 5, we get the following constraint on $\mathbf{a}$:

$$\begin{aligned} (\mathbf{M} \oplus \mathbf{I}_n) \cdot \mathbf{a} = (\mathbf{M}^2 \oplus \mathbf{I}_n) \cdot \mathbf{a} &\implies \mathbf{M}^2 \cdot \mathbf{a} = \mathbf{M} \cdot \mathbf{a} \\ &\implies \mathbf{M} \cdot (\mathbf{M} \cdot \mathbf{a}) = \mathbf{M} \cdot \mathbf{a} \end{aligned} \tag{6}$$

Hence, $\mathbf{M} \cdot \mathbf{a}$ must be a fixed point of $\mathbf{M}$. If $\mathbf{M} \oplus \mathbf{I}_n$ is invertible, then $\mathbf{a} = \mathbf{b}$ should also be satisfied, because of Eqn. 5. If extra swapping reduces the number of fixed points and increases the minimum weight of the fixed points, then corresponding diffusion layer is better to be used in a hash function than using the involutory matrix.

4 Efficiently Implementable F Function Families

While generating binary diffusion layer matrices in a Feistel structure, we tried two sets of efficiently implementable $\mathbf{M}$ matrices. Depending on the implementation scenario, $\mathbf{M}$ matrices can be used as look up tables or functional units. We investigated only matrices with power of two input bits.

Since we consider both software and hardware implementations, $\mathbf{M}$ matrices are chosen suitable for both environments. In hardware, number of bits XORed to calculate output bits is relevant for area and speed performance. Hence, for hardware efficiency, low number of XORs for each output bit is preferred. In software on the other hand, instruction set determines the set of efficiently implementable linear layers. We only considered bit-wise shift, rotate, and XOR operations since these are common instructions in most CPUs.

4.1 Matrix family based on rotation

Definition 2 (Rotational XOR matrices)**.** *Let n be a positive integer and $U \subset \{0, 1, \ldots, n-1\}$ be a nonempty set. The rotational XOR matrix (RX for short) determined by U over n-bit vectors is denoted by $\mathbf{RX}_n^U$ and defined as follows:*

$$\mathbf{RX}_n^U \cdot \mathbf{a} = \bigoplus_{i \in U} (\mathbf{a} \lll i)$$

where $\lll$ is bit-wise left cyclic shift (rotation) operation on n-bit vectors.

If $|U| = u$, $BN(\mathbf{RX}_n^U) \leq u + 1$. Not all U satisfy the equality, but some do as we saw in our experiments. Hence, using an RX matrix with u rotations, three-round Feistel scheme can produce matrices with BN at most $2(u + 1)$. Some index set U cannot achieve the $2(|U| + 1)$ bound. Following Lemma and Theorem categorise one such family:

Lemma 1. *Let n, U, and $\mathbf{M} = \mathbf{RX}_n^U$ be defined as in Definition 2. Then,*

$$\mathbf{M}^2 \cdot \mathbf{a} = \bigoplus_{i \in U} (\mathbf{a} \lll 2i)$$

and;

$$\mathbf{M}^3 \cdot \mathbf{a} = \bigoplus_{(i,j) \in U \times U} (\mathbf{a} \lll (2i + j))$$

where rotation amounts are calculated in $\mathbb{Z}_n$, ring of integers modulo n.

Theorem 4. *Let $\mathbf{M} = \mathbf{RX}_n^U$, $V = \{0\} \cup \{2i \mod n | i \in U\}$. If $|V| < |U| + 1$, then $BN(\mathbf{D_M}) \leq 2|U|$.*

Proof. We use the notation in Eqn. 2 of Section 3.1. Let $\mathbf{a} = \mathbf{0}$ and $|\mathbf{b}| = 1$, i.e. a vector with one single bit on the right hand side of Figure 1 is set to one, and others to zero. Then by Lemma 1, the output is:

$$\begin{bmatrix} \bigoplus_{i \in U} (\mathbf{b} \lll i) \\ \mathbf{b} \oplus \bigoplus_{i \in U} (\mathbf{b} \lll 2i) \end{bmatrix} \tag{7}$$

where $|\bigoplus_{i \in U} (\mathbf{b} \lll i)| = |U|$ by definition. Since $|V| < |U| + 1$ by assumption, some indices are equal in the unordered list consisting of zero and double (in $\mathbb{Z}_n$) of each index in U. XORing a vector's rotated values with the same rotation amount cancels each other. Therefore $t = |\mathbf{b} \oplus \bigoplus_{i \in U} (\mathbf{b} \lll 2i)| \leq |U| - 1$ which means the sum of input-output Hamming weight satisfies $1 + |U| + t \leq 2|U|$. □

For an RX matrix $\mathbf{M}$ satisfying the maximal BN of $|U| + 1$, any column and row has weight $|U|$. So the condition in the proof of Theorem 1 is satisfied. Therefore, two round Feistel instead of three may generate a diffusion layer with BN at most $|U| + 2$.

The linear BN of an RX matrix is the same with its differential BN. This fact is proven by Lemma 2, Lemma 3 and Theorem 5. Before giving the lemmas and the theorem, the *reverse matrix* should be defined:

Definition 3 (Reverse Matrix)**.** *Let n be a positive integer. $n \times n$ reverse matrix is denoted by $\mathbf{R}_n$ and defined as the binary anti-diagonal matrix with all 1 bits on the anti-diagonal and zero in other positions.*

Reverse matrix reverses the order of bits in an n-bit vector, and it is an involution. If reverse matrix is multiplied from left, it reverses the order of the rows, otherwise it reverses the order of columns. Using $\mathbf{R}_n$, $\mathbf{R}_{2n}$ can be written as follows:

$$\mathbf{R}_{2n} = \begin{bmatrix} \mathbf{0}_n & \mathbf{R}_n \\ \mathbf{R}_n & \mathbf{0}_n \end{bmatrix}$$

For an $n \times n$ binary matrix $\mathbf{M}$, let $\mathbf{M}[r, c]$ denote the entry in r^{th} row and c^{th} column, where $0 \leq r, c \leq n-1$. In this notation, reversing is equivalent to inverting the indices additively modulo n, i.e., index i becomes $-i$ in $\mathbb{Z}_n$.

Lemma 2. *Let n be a positive integer, $U = \{i\}$ for some $i \in \{0, 1, \ldots, n-1\}$, and $\mathbf{M} = \mathbf{RX}_n^U$. Then $\mathbf{M}^T = \mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n$, where $\mathbf{M}^T$ denotes the transpose of the matrix $\mathbf{M}$.*

Proof. There are n one bits in $\mathbf{M}$, and since reversing the columns and rows do not change the number of 1's, we will only consider the position of 1's to show the matrix equality. By definition, $\mathbf{M}[r, r+i] = 1, \forall r \in \mathbb{Z}_n$, and other positions are zero. All index arithmetic will be done in $\mathbb{Z}_n$, and for the sake of simplicity, $\forall r \in \mathbb{Z}_n$ part will be omitted in the following matrix definitions. By the definition of transpose operation, $\mathbf{M}^T[r+i, r] = 1$. Call $\overline{\mathbf{M}} = \mathbf{M} \cdot \mathbf{R}_n$. Since $\overline{\mathbf{M}}$ is the column reversed version of $\mathbf{M}$, we have $\overline{\mathbf{M}}[r, -r-i] = 1$. Similarly, if $\overline{\overline{\mathbf{M}}} = \mathbf{R}_n \cdot \overline{\mathbf{M}}$, then $\overline{\overline{\mathbf{M}}}[-r, -r-i] = 1$. We need to show that $\mathbf{M}^T = \overline{\overline{\mathbf{M}}}$, which is equivalent to saying that $A = \{(r+i, r) | r \in \mathbb{Z}_n\} = \{(-r, -r-i) | r \in \mathbb{Z}_n\} = B$. Since the cardinality of the sets A and B are the same, it is enough to show that $\forall r_1 \in \mathbb{Z}_n, \exists r_2 \in \mathbb{Z}_n$ such that $(r_1 + i, r_1) = (-r_2, -r_2 - i)$. Let $r_1 \in \mathbb{Z}_n$ and define $r_2 = -r_1 - i \in \mathbb{Z}_n$. Then, left and right indices are equal. Therefore $A = B$, which completes the proof. □

Lemma 3. *For any positive integer n and nonempty $U \subset \{0, 1, \ldots, n-1\}$, $(\mathbf{RX}_n^U)^T = \mathbf{R}_n \cdot \mathbf{RX}_n^U \cdot \mathbf{R}_n$.*

Proof. This follows directly from Lemma 2 and the distribution property of transpose over XOR operation, i.e., $(\mathbf{A} \oplus \mathbf{B})^T = \mathbf{A}^T \oplus \mathbf{B}^T$ for any binary matrices $\mathbf{A}, \mathbf{B}$. □

Theorem 5. *Let n be a positive integer, $U \subset \{0, 1, \ldots, n-1\}$ be a nonempty set, and $\mathbf{M} = \mathbf{RX}_n^U$. If $\mathbf{D}_{\mathbf{M}}$ is the three-round Feistel scheme with round function $\mathbf{M}$, then $BN(\mathbf{D}_{\mathbf{M}}) = BN(\mathbf{D}_{\mathbf{M}}{}^T)$.*

Proof. Following the notation in Eqn. (1) in Section 3.1, transpose of $\mathbf{D_M}$ is:

$$\mathbf{D_M}^T = \begin{bmatrix} \mathbf{M}^2 \oplus \mathbf{I}_n & \mathbf{M} \\ \mathbf{M}^3 & \mathbf{M}^2 \oplus \mathbf{I}_n \end{bmatrix}^T = \begin{bmatrix} (\mathbf{M}^T)^2 \oplus \mathbf{I}_n & (\mathbf{M}^T)^3 \\ \mathbf{M}^T & (\mathbf{M}^T)^2 \oplus \mathbf{I}_n \end{bmatrix}$$

By Lemma 3, we have $\mathbf{M}^T = \mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n$. Then,

$$\mathbf{D_M}^T = \begin{bmatrix} (\mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n)^2 \oplus \mathbf{I}_n & (\mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n)^3 \\ \mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n & (\mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n)^2 \oplus \mathbf{I}_n \end{bmatrix}$$

Clearly, $(\mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n)^k = \mathbf{R}_n \cdot \mathbf{M}^k \cdot \mathbf{R}_n$, since $\mathbf{R}_n^{-1} = \mathbf{R}_n$. Therefore,

$$\mathbf{D_M}^T = \begin{bmatrix} \mathbf{R}_n \cdot \mathbf{M}^2 \cdot \mathbf{R}_n \oplus \mathbf{I}_n & \mathbf{R}_n \cdot \mathbf{M}^3 \cdot \mathbf{R}_n \\ \mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n & \mathbf{R}_n \cdot \mathbf{M}^2 \cdot \mathbf{R}_n \oplus \mathbf{I}_n \end{bmatrix}$$

Now consider $\mathbf{R}_{2n} \cdot \mathbf{D_M} \cdot \mathbf{R}_{2n}$;

$$\mathbf{R}_{2n} \cdot \mathbf{D_M} \cdot \mathbf{R}_{2n} = \mathbf{R}_{2n} \cdot \begin{bmatrix} \mathbf{M}^2 \oplus \mathbf{I}_n & \mathbf{M} \\ \mathbf{M}^3 & \mathbf{M}^2 \oplus \mathbf{I}_n \end{bmatrix} \cdot \mathbf{R}_{2n} =$$
$$\begin{bmatrix} \mathbf{R}_n \cdot (\mathbf{M}^2 \oplus \mathbf{I}_n) \cdot \mathbf{R}_n & \mathbf{R}_n \cdot \mathbf{M}^3 \cdot \mathbf{R}_n \\ \mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n & \mathbf{R}_n \cdot (\mathbf{M}^2 \oplus \mathbf{I}_n) \cdot \mathbf{R}_n \end{bmatrix} = \mathbf{D_M}^T$$

Hence, $\mathbf{D_M}^T = \mathbf{R}_{2n} \cdot \mathbf{D_M} \cdot \mathbf{R}_{2n}$. Since $\mathbf{R}_{2n}^{-1} = \mathbf{R}_{2n}$, $\mathbf{D_M}$ and $\mathbf{D_M}^T$ are similar matrices and produce equivalent codes when used as non-systematic part of the generator matrices in systematic form [3]. It is a well known fact that their BN are the same [23]. □

Because of Theorem 5, we did not need to check the BN of the transpose of the matrix to calculate the linear BN. Since BN calculations are the most time consuming part, this reduces the analysis time almost by half. Moreover, this also gives the same bounds of provable minimum number of active S-boxes in both differential and linear characteristics.

4.2 Matrix family based on shifting

Definition 4 (Shift XOR matrices)**.** *Let n be a positive integer, U and V be two subsets of $\{0, 1, \ldots, n-1\}$. The shift XOR matrix (SX for short) determined by U and V over n-bit vectors is denoted by $\mathbf{SX}_n^{U,V}$ and defined as follows:*

$$\mathbf{SX}_n^{U,V} \cdot \mathbf{a} = \bigoplus_{i \in U} (\mathbf{a} \ll i) \oplus \bigoplus_{j \in V} (\mathbf{a} \gg j)$$

where $\ll$ and $\gg$ denote bit-wise shift operations to the left and right, respectively.

Each shift operation has a corresponding left multiplier matrix. Since shifted values are XORed, this corresponds to XOR of matrices. After a shift by t operation on a vector, t bits on the shift direction is lost and t zero bits are fed in the opposite direction. In matrix notation, this means t columns are zero in

the corresponding matrix. For each left shift index l_i, a '1' bit is set in column l_i of row zero ($\mathbf{M}[0, l_i]$). The position of the '1' bit is moved to the right for each consecutive, until reaching rightmost column. So a left shift corresponds to an upper triangular matrix where '1' bits are placed parallel to the diagonal. For right shift index r_i, row r_i of column zero ($\mathbf{M}[r_i, 0]$) is set to '1', and its place is moved to the right for each consecutive row, until reaching bottom row. Similar to left shift matrix, right shift matrices produce a lower triangular matrix with '1' bits placed parallel to the diagonal. An example for $n = 8$, $U = \{3, 4\}$, and $V = \{2, 6\}$ is given in Figure 2.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 2: $\mathbf{SX}_8^{U,V}$ for $U = \{3, 4\}, V = \{2, 6\}$.

For an SX matrix $\mathbf{M}$, $\mathbf{M}^T = \mathbf{R}_n \cdot \mathbf{M} \cdot \mathbf{R}_n$. This can be proven by similar arguments as in Lemma 2 and Lemma 3. Hence, Theorem 5 also applies to diffusion layers constructed by SX matrices, which means linear and differential BNs are the same for a three-round Feistel scheme with an SX round function.

Number of ones in the left most column of an SX matrix is determined by the number of right shifts. Similarly, number of ones in the rightmost column is determined by the number of left shifts. Since minimum weight of the columns put an upper bound on BN, it is a good idea to select the direction sets cardinality $|U|$ and $|V|$ equal or close.

Assume that there are m left shifts and m right shifts for an SX matrix $\mathbf{M}$. Also assume that if exists, zero shifting is only included in one of the sets (otherwise they cancel each other). To make the other columns than the first and last one to have weights greater than or equal to m, following property should be satisfied:

$$\max\{v \in V\} + \min\{u \in U\} \leq n \tag{8}$$

This is because of the fact that maximum term in right shift ($\overline{v}$) vanishes first while moving from first to last column. It vanishes in the $(n - \overline{v})^{th}$ column. If minimum term of the left shift ($\overline{u}$) is greater than $n - \overline{v}$, then there would be some columns with less than m ones. Similarly, for keeping at least m ones in each row the following inequality is required:

$$\max\{u \in U\} + \min\{v \in V\} \leq n \tag{9}$$

In Section 5.2, we used this idea while generating diffusion layers based on SX matrices.

5 Experimental Analysis

We used RX and SX matrices as Feistel F functions in our experiments. Bit size of the F function in three-round Feistel is denoted by n, and chosen such that $n \in \{8, 16, 32\}$ (which are typical word sizes for general purpose CPUs) to generate 16, 32, and 64-bit diffusion layers. For each resultant diffusion layer matrix, we calculated the BN of the matrix using `magma` computer algebra system version `2.21-7`.

5.1 Experiments on RX matrices

For RX matrices, we chose rotation set sizes $|U| \in \{2, 3, 4, 5, 6, 7\}$. If number of subsets of size $|U|$ is of reasonable size (i.e. computation finishes at most in a few hours in a standard PC), we exhaustively searched all possible rotation sets U.

To reduce execution time of our exhaustive search algorithms, we used Theorem 2, 4, and 5. We applied the test defined in Theorem 4 before trying any index set. This test reduces search space, but the actual reduction depends on n and $|U|$. For some parameters, this test reduced the total time to almost one thirds.

If index set passed the previous test, we tested for the BN of the small matrix if its BN is $|U|+1$, since Theorem 2 stated that only this matrices may generate diffusion layers with BN equal to $2(|U| + 1)$. Theorem 5 let us to calculate the BN for only $\mathbf{D_M}$, not the transpose. Experimental results for diffusion layers using RX matrices are summarised in Table 2.

Some examples of RX matrices which give the maximal BN in our construction are given in Appendix A together with the dimension of the kernel of small matrices. If known maximal BN is attained for $|U|$ rotations, we did not present results for more than $|U|$ rotations since they are less efficient. For example, $n = 8, |U| = 4$ satisfies $BN(\mathbf{D_M}) = 8$, i.e. $\mathbf{D_M}$ is a MDBL code generator, hence we did not search for $|U| > 4$ case for $n = 8$.

5.2 Experiments on SX matrices

While analysing SX matrix based diffusion layers, we first fixed the sum $s = |U| + |V|$, and then divide s into two subset sizes $u = |U|$ and $v = |V|$ in a balanced way for $s > 6$ (we tried all partitions for $s \leq 6$). If $s = 2m$ for some integer m, then we selected $u = v = m$. If $s = 2m + 1$ for some integer m, we tried the cases $u = m + 1, v = m$ and $u = m, v = m + 1$.

Table 2: Experimental results on BN of proposed method using RX matrices. **Bold** and *italic* numbers represent the achievement to the known and achievable maximal BN for that $n, |U|$ combination respectively. Fourth column shows the number of $\mathbf{D_M}$ matrices with maximal BN.

n	$\|U\|$	Max $BN(\mathbf{D_M})$	$\#\mathbf{D_M}$ with max. BN
8	2	*6*	8
8	3	6	40
8	4	**8**	8
16	2	*6*	68
16	3	*8*	240
16	4	*10*	224
16	5	**12**	48
32	2	*6*	380
32	3	*8*	3584
32	4	*10*	18896
32	5	*12*	†
32	6	*14*	†
32	7	*16*	†

† : Exhaustive search did not finished in reasonable time

We searched for $s \in \{2, 3, \ldots, 9\}$. Shifting by zero is not shifting at all, and if both sets U, V include zero, they cancel each other because of the XOR operation. Hence we excluded zero from subset U and V. For each subset selection, we analysed both cases with and without zero shifting. For $s \in \{2, 3, \ldots, 6\}$, the matrices with highest BN are the ones with XORing the zero shifted value. For $s > 6$, we only tried this case.

Table 3 shows highest BN values we found for different s and n values. s denotes the total number of shifts and do not include the zero shifting. We excluded results of $s = 9$ since it did not produce better results than $s = 8$ case. We also excluded results for some s values if it did not increased the maximum BN of previous s value.

For $s = 2m$, best achievable BN is calculated as follows : Since m left shifts, m right shifts and non-shifted original input are XORed, there are $m+1$ ones in the first and the last column (note that in Figure 2, XORing with non-shifted value is not included). Therefore, maximal BN for small matrix is at most $m + 2$. By Theorem 2, maximum achievable BN for generated diffusion layer is $2m + 4 = s + 4$.

Some efficient examples of SX matrices are given in Appendix B with the kernel dimension of the small matrix. Experiments showed that the size of the kernel is small for the best matrices found. Some of the matrices has zero

Table 3: Experimental results on BN of proposed method using SX matrices. **Bold** and *italic* numbers represent the achievement to the known and achievable maximal BN respectively. Fourth column shows the number of $\mathbf{D_M}$ matrices with maximal BN.

n	s	Max. BN	# $\mathbf{D_M}$ with max. BN
8	2	5	1
8	3	*6*	2
8	6	7	5
8	8	**8**	4
16	2	5	2
16	3	*6*	62
16	4	*8*	20
16	6	*10*	175
32	2	5	5
32	3	*6*	590
32	4	*8*	568
32	6	*10*	†
32	8	*12*	†

† : Exhaustive search did not finished in reasonable time

dimensional kernel. This is the best achievable case in terms of minimising number of fixed points in generated diffusion layer. Some matrices do not satisfy the properties in equations (8) and (9) and they do not achieve the highest BN possible. We gave those matrices since highest possible BN could not be achieved and they were the best in terms of BN in their search space.

6 Implementation Complexity

Hardware and software area of RX and SX matrices depending on subsets U and V can be estimated easily. For RX matrices, each output bit is the XOR of $|U|$ many input bits. Since RX matrices are circulant, it is possible to make optimisations using common XOR outputs. Moreover, it is also possible to implement a single row of the matrix and use the input by rotating if an extra register can be utilised. In this method, n cycles are necessary for a single Feistel round. In software on the other hand, there are at most $|U|$ rotation and exactly $|U| - 1$ XORs for an application of $\mathbf{M}$. Complexity of a single rotation depend on the instruction set of the CPU. Some CPUs have parametric rotation instructions where each rotation takes only one instruction. If only parametric shift is available, two shifts and one OR/XOR is sufficient to complete a single rotation. If only a single bit shift is available, then the performance depends on

the bit size n.

In SX case, number of XOR inputs for each out bit depends on both subsets U and V. There are $|U| + |V|$ shifts and $|U| + |V|$ XORs in our generated matrices. In software, if there is no parametric rotation instruction is available, an RX matrix with r rotations is slightly less efficient than an SX matrix with $s = 2r - 2$ or $s = 2r - 1$ shifts. This is due to the fact that rotations can be emulated by two shifts and an XOR. In fact, in both cases, shift based matrices have less instruction count when zero rotation is not used in rotational matrix. By the use of Theorem 4, this is already the case in tested rotational matrices satisfying maximal achievable BN. If $0 \in U$, then $0 \in \{2i | i \in U\}$ and this cancels the y in Eqn. 7, and that matrix cannot generate a diffusion layer with BN equal to maximum achievable bound. Since these matrices have the same maximal BN, SX is preferable over RX if a tuple of index sets satisfy best achievable BN. If only single bit shift is available in the CPU, then shift sets with smaller values are more efficient.

A linear diffusion layer may also be implemented as look-up table in software [13]. In such a case, only the table of $\mathbf{M}$ would be necessary for our construction. So the total area complexity is $n2^n$-bits. To complete three rounds of Feistel, three table look-ups, three XORs and a final swap is necessary (swap may be omitted for non-involution case). On the other hand, for a random $2n$-bit linear layer, four tables of $n2^n$-bits is necessary if the linear layer matrix is divided into four equal square sub-matrices. If the matrix is not an involution or does not have a simple inverse permitting the use of the same tables, four extra tables of $n2^n$-bits may be also necessary. For both linear layer and its inverse, time complexity is at least four table look-ups and four XORs. Assuming that time complexity of a swap is close to one table look-up and one XOR, time complexities are close for the proposed method and random linear layer. Therefore, our method can be used to reduce the table size by a factor of $\frac{1}{4}$ without significant performance lost for this kind of implementation. It is possible to further decrease area by partitioning the original matrix into smaller sub matrices, but this method increases the number of table look-up and XOR count.

Comparison with some direct constructions : It is possible to design a diffusion layer as an RX or SX matrix directly without using the Feistel scheme. If this diffusion layer is to be used in an SPN cipher (such as a bitslice cipher), the matrix should also be invertible. Moreover, if encryption and decryption is implemented in the same platform, the area overhead of the inverse matrix should also be considered.

We made experiments on RX matrices and saw that the bounds on maximal branch numbers could be achieved by direct constructions. For example, to get a 16-bit diffusion layer with BN=8, we needed $|U| = 7$, i.e 7 rotated vales XORed together. This is the same BN when we utilised 8-bit RX matrix in Feistel diffusion layer with $|U| = 4$. On the other hand, none of the direct constructions were involution. Direct construction may be more efficient in some implementation scenarios, but the matrix need not to be an involution,

that is decryption overhead may be significant if used in a block cipher.

7 Conclusion and Future Work

In this work, we propose a new generic method for creating efficiently implementable binary matrices with high branch number and reasonable number of fixed points. We use three round Feistel scheme with a linear round function, so generated matrices are self inverse, i.e. involution. Hence, there is no overhead for the inverse diffusion layer when used in an SPN-like block cipher with encryption/decryption implementation. We define and use two family of efficiently implementable linear mappings as Feistel round function. Some of the matrices we found are MDBL, and some have known maximal BN. The matrices we have generated are especially efficient to use in bitslice block ciphers and hash functions. Our matrices are suitable for both software and hardware, and can be implemented efficiently even in low cost CPUs.

Future work : Our work can be generalised in many ways. In Feistel round, other families of linear functions, or even random look-up tables can be used. Different but related (e.g. by rotation) F functions can be used in each round. Generalised Feistel schemes with more rounds can be used to generate matrices with greater bit size. Some rotation subsets in RX matrices generated the maximal achievable BN for RX matrix itself and generated diffusion layer. If formulae for arithmetic relations in these subsets can be found, then matrices with higher bit size and BN can be generated without trying to calculate BN by exhaustive search. Direct RX and SX construction as diffusion layer with simple inverse (such as rotating input/output) can be useful in some implementation mediums. Our matrices can also be used as binary linear code, if efficient decoding algorithms can be found.

Acknowledgements Authors would like to thank to Ferhat Karakoç, Oğuzhan Ersoy, Güven Yücetürk, Mehmet E. Gönen, and Muhammet F. Esgin for their helpful comments on the writing of this paper.

References

[1] Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçin, T.: Block ciphers - focus on the linear layer (feat. PRIDE). In: J.A. Garay, R. Gennaro (eds.) Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I, *Lecture Notes in Computer Science*, vol. 8616, pp. 57–76. Springer (2014). DOI 10.1007/978-3-662-44371-2_4. URL http://dx.doi.org/10.1007/978-3-662-44371-2_4

[2] Aslan, B., Sakalli, M.T.: Algebraic construction of cryptographically good binary linear transformations. Security and Communi-

cation Networks **7**(1), 53–63 (2014). DOI 10.1002/sec.556. URL http://dx.doi.org/10.1002/sec.556

[3] Augot, D., Finiasz, M.: Direct construction of recursive MDS diffusion layers using shortened BCH codes. In: Cid and Rechberger [9], pp. 3–17. DOI 10.1007/978-3-662-46706-0_1. URL http://dx.doi.org/10.1007/978-3-662-46706-0_1

[4] Baysal, A., Şahin, S.: Roadrunner: A small and fast bitslice block cipher for low cost 8-bit processors. IACR Cryptology ePrint Archive **2015**, 906 (2015). URL http://eprint.iacr.org/2015/906

[5] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Keccak. In: T. Johansson, P.Q. Nguyen (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings, *Lecture Notes in Computer Science*, vol. 7881, pp. 313–314. Springer (2013). DOI 10.1007/978-3-642-38348-9_19. URL http://dx.doi.org/10.1007/978-3-642-38348-9_19

[6] Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: A. Menezes, S.A. Vanstone (eds.) Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings, *Lecture Notes in Computer Science*, vol. 537, pp. 2–21. Springer (1990). DOI 10.1007/3-540-38424-3_1. URL http://dx.doi.org/10.1007/3-540-38424-3_1

[7] Canteaut, A.: Similarities between encryption and decryption: How far can we go? In: Selected Areas in Cryptography-SAC 2013. Springer (2013)

[8] Canteaut, A., Duval, S., Leurent, G.: Construction of lightweight s-boxes using feistel and MISTY structures (full version). IACR Cryptology ePrint Archive **2015**, 711 (2015). URL http://eprint.iacr.org/2015/711

[9] Cid, C., Rechberger, C. (eds.): Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 8540. Springer (2015). DOI 10.1007/978-3-662-46706-0. URL http://dx.doi.org/10.1007/978-3-662-46706-0

[10] Daemen, J.: Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis. K.U.Leuven (1995). http://jda.noekeon.org/

[11] Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer Verlag, Berlin, Heidelberg, New York (2002)

[12] Grassl, M.: Codetables.de. http://www.codetables.de/ (2016). Accessed: 2016-01-04

[13] Grosso, V., Leurent, G., Standaert, F., Varici, K.: Ls-designs: Bitslice encryption for efficient masked software implementations. In: Cid and Rechberger [9], pp. 18–37. DOI 10.1007/978-3-662-46706-0_2. URL http://dx.doi.org/10.1007/978-3-662-46706-0_2

[14] Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight hash functions. In: P. Rogaway (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings, *Lecture Notes in Computer Science*, vol. 6841, pp. 222–239. Springer (2011). DOI 10.1007/978-3-642-22792-9_13. URL http://dx.doi.org/10.1007/978-3-642-22792-9_13

[15] Guo, Z., Wu, W., Gao, S.: Constructing Lightweight Optimal Diffusion Primitives with Feistel Structure, pp. 352–372. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-31301-6_21. URL http://dx.doi.org/10.1007/978-3-319-31301-6_21

[16] Gupta, K.C., Ray, I.G.: On constructions of involutory MDS matrices. In: A. Youssef, A. Nitaj, A.E. Hassanien (eds.) Progress in Cryptology - AFRICACRYPT 2013, 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings, *Lecture Notes in Computer Science*, vol. 7918, pp. 43–60. Springer (2013). DOI 10.1007/978-3-642-38553-7_3. URL http://dx.doi.org/10.1007/978-3-642-38553-7_3

[17] Gupta, K.C., Ray, I.G.: On constructions of MDS matrices from companion matrices for lightweight cryptography. In: A. Cuzzocrea, C. Kittl, D.E. Simos, E.R. Weippl, L. Xu, A. Cuzzocrea, C. Kittl, D.E. Simos, E.R. Weippl, L. Xu (eds.) Security Engineering and Intelligence Informatics - CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings, *Lecture Notes in Computer Science*, vol. 8128, pp. 29–43. Springer (2013). DOI 10.1007/978-3-642-40588-4_3. URL http://dx.doi.org/10.1007/978-3-642-40588-4_3

[18] Helleseth, T. (ed.): Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings, *Lecture Notes in Computer Science*, vol. 765. Springer (1994)

[19] Hong, S., Lee, S., Lim, J., Sung, J., Cheon, D.H., Cho, I.: Provable security against differential and linear cryptanalysis for the SPN structure. In: B. Schneier (ed.) Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings, *Lecture Notes in Computer Science*, vol. 1978, pp. 273–283. Springer (2000). DOI 10.1007/3-540-44706-7_19. URL http://dx.doi.org/10.1007/3-540-44706-7_19

[20] Koo, B., Jang, H.S., Song, J.H.: Constructing and cryptanalysis of a 16 × 16 binary matrix as a diffusion layer. In: K. Chae, M. Yung (eds.) Information Security Applications, 4th International Workshop, WISA 2003, Jeju

Island, Korea, August 25-27, 2003, Revised Papers, *Lecture Notes in Computer Science*, vol. 2908, pp. 489–503. Springer (2003). DOI 10.1007/978-3-540-24591-9_36. URL http://dx.doi.org/10.1007/978-3-540-24591-9_36

[21] Kwon, D., Kim, J., Park, S., Sung, S.H., Sohn, Y., Song, J.H., Yeom, Y., Yoon, E., Lee, S., Lee, J., Chee, S., Han, D., Hong, J.: New block cipher: ARIA. In: J.I. Lim, D.H. Lee (eds.) Information Security and Cryptology - ICISC 2003, 6th International Conference, Seoul, Korea, November 27-28, 2003, Revised Papers, *Lecture Notes in Computer Science*, vol. 2971, pp. 432–445. Springer (2003). DOI 10.1007/978-3-540-24691-6_32. URL http://dx.doi.org/10.1007/978-3-540-24691-6_32

[22] Lai, X., Massey, J.L.: A proposal for a new block encryption standard. In: I. Damgård (ed.) Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings, *Lecture Notes in Computer Science*, vol. 473, pp. 389–404. Springer (1990). URL http://link.springer.de/link/service/series/0558/bibs/0473/047 30389.htm

[23] Ling, S., Xing, C.: Coding theory: a first course. Cambridge University Press (2004)

[24] Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth [18], pp. 386–397. DOI 10.1007/3-540-48285-7_33. URL http://dx.doi.org/10.1007/3-540-48285-7_33

[25] Matsui, M.: New block encryption algorithm MISTY. In: E. Biham (ed.) Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings, *Lecture Notes in Computer Science*, vol. 1267, pp. 54–68. Springer (1997). DOI 10.1007/BFb0052334. URL http://dx.doi.org/10.1007/BFb0052334

[26] Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseth [18], pp. 55–64. DOI 10.1007/3-540-48285-7_6. URL http://dx.doi.org/10.1007/3-540-48285-7_6

[27] Sajadieh, M., Dakhilalian, M., Mala, H., Omoomi, B.: On construction of involutory MDS matrices from vandermonde matrices in GF(2 q). Des. Codes Cryptography **64**(3), 287–308 (2012). DOI 10.1007/s10623-011-9578-x. URL http://dx.doi.org/10.1007/s10623-011-9578-x

[28] Sajadieh, M., Dakhilalian, M., Mala, H., Sepehrdad, P.: Efficient recursive diffusion layers for block ciphers and hash functions. J. Cryptology **28**(2), 240–256 (2015). DOI 10.1007/s00145-013-9163-8. URL http://dx.doi.org/10.1007/s00145-013-9163-8

[29] Sakalli, M.T., Aslan, B.: On the algebraic construction of cryptographically good 32×32 binary linear transformations. J. Computational Applied

Mathematics **259**, 485–494 (2014). DOI 10.1016/j.cam.2013.05.008. URL http://dx.doi.org/10.1016/j.cam.2013.05.008

[30] Sim, S.M., Khoo, K., Oggier, F.E., Peyrin, T.: Lightweight MDS involution matrices. In: G. Leander (ed.) Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 9054, pp. 471–493. Springer (2015). DOI 10.1007/978-3-662-48116-5_23. URL http://dx.doi.org/10.1007/978-3-662-48116-5_23

[31] of Standards, N.I., Technology: Nist computer security division - the sha-3 cryptographic hash algorithm competition, november 2007 - october 2012. http://csrc.nist.gov/groups/ST/hash/sha-3/ (2016). Accessed: 2016-01-05

[32] Wu, H.: The hash function jh. Submission to NIST (round 3) (2011)

[33] Wu, S., Wang, M., Wu, W.: Recursive diffusion layers for (lightweight) block ciphers and hash functions. In: L.R. Knudsen, H. Wu (eds.) Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 7707, pp. 355–371. Springer (2012). DOI 10.1007/978-3-642-35999-6_23. URL http://dx.doi.org/10.1007/978-3-642-35999-6_23

[34] Z'aba, M.R.: Analysis of linear relationships in block ciphers. Ph.D. thesis, Queensland University of Technology (2010). URL http://eprints.qut.edu.au/35725/

A RX Matrices Satisfying Maximal BN

A.1 $n = 8, |U| = 2, BN = 6$

In the following table, we give both the rotation subsets U and the first row of the RX matrix. RX matrices are circulant where the next row is 1-bit right rotated version of the previous one. The kernel dimension of each matrix is one, i.e there are $2^1 = 2$ vector in the null space of RX matrices.

U	First row	U	First row
{1,2}	[01100000]	{2,7}	[00100001]
{1,6}	[01000010]	{3,6}	[00010010]
{2,3}	[00110000]	{5,6}	[00000110]
{2,5}	[00100100]	{6,7}	[00000011]

A.2 $n = 8, |U| = 4, BN = 8$ (MDBL matrices)

Kernel dimension of each RX matrix in the following table is one.

U	First row	U	First row
{1,2,3,5}	[01110100]	{1,3,6,7}	[01010011]
{1,2,3,7}	[01110001]	{1,5,6,7}	[01000111]
{1,2,5,7}	[01100101]	{2,3,5,7}	[00110101]
{1,3,5,6}	[01010110]	{3,5,6,7}	[00010111]

A.3 $n = 16, |U| = 5, BN = 12$ (MDBL matrices)

In the following table we give only rotation subset U for all 48 matrices we have found. All kernels are zero dimensional, i.e., only zero is in the kernel.

{1,2, 3, 5,14}	{1,2, 3, 7,14}	{1, 2, 4, 7,14}	{1, 2, 5, 7,14}
{1,2, 7,11,14}	{1,2, 7,12,14}	{1, 2, 7,13,14}	{1, 2,11,13,14}
{1,3, 5, 6,10}	{1,3, 6, 7,10}	{1, 4, 6, 7,10}	{1, 5, 6, 7,10}
{1,6, 7,10,11}	{1,6, 7,10,12}	{1, 6, 7,10,13}	{1, 6,10,11,13}
{2,3, 4, 5,14}	{2,3, 5, 7,14}	{2, 3, 5, 9,14}	{2, 3, 5,12,14}
{2,3, 5,14,15}	{2,3, 9,14,15}	{2, 4, 9,14,15}	{2, 4,11,13,14}
{2,5, 9,14,15}	{2,7,11,13,14}	{2, 9,11,13,14}	{2, 9,11,14,15}
{2,9,12,14,15}	{2,9,13,14,15}	{2,11,12,13,14}	{2,11,13,14,15}
{3,4, 5, 6,10}	{3,5, 6, 7,10}	{3, 5, 6, 9,10}	{3, 5, 6,10,12}
{3,5, 6,10,15}	{3,6, 9,10,15}	{4, 6, 9,10,15}	{4, 6,10,11,13}
{5,6, 9,10,15}	{6,7,10,11,13}	{6, 9,10,11,13}	{6, 9,10,11,15}
{6,9,10,12,15}	{6,9,10,13,15}	{6,10,11,12,13}	{6,10,11,13,15}

B SX Matrices Satisfying Maximal BN

For each matrix in this section, non-shifted vector is XORed to the shifted values. For a total of s shifts, $s + 1$ values are XORed together in the SX matrix. For example, for $U = V = \{1\}$, resultant SX matrix is:

$$\mathbf{M} \cdot \mathbf{a} = \mathbf{a} \oplus (\mathbf{a} \ll 1) \oplus (\mathbf{a} \gg 1)$$

n	s	BN	# of matrices	U	V	Kernel dimn.
8	2	5	1	{1}	{1}	1
8	3	6	2	{1,2}	{2}	0
				{2}	{1,2}	0
8	6	7	5	{1,2,3,7}	{3,4}	0
				{1,2,4}	{1,4,7}	0
				{1,2,5}	{1,2,5}	0
				{1,4,7}	{1,2,4}	0
				{3,4}	{1,2,3,7}	0
8	8	8	4	{1,3,4,5}	{3,4,5,7}	0
				{1,3,4,7}	{1,4,5,7}	0
				{1,4,5,7}	{1,3,4,7}	0
				{3,4,5,7}	{1,3,4,5}	0
16	2	5	2	{1}	{1}	0
				{2}	{2}	2
32	2	5	5	{1}	{1}	1
				{2}	{2}	0
				{3}	{3}	2
				{4}	{4}	4
				{5}	{5}	0