IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Efficient methods to generate cryptographically significant binary diffusion layers

ISSN 1751-8709
Received on 16th February 2016
Revised 30th June 2016
Accepted on 26th July 2016
E-First on 9th September 2016
doi: 10.1049/iet-ifs.2016.0085
www.ietdl.org

Sedat Akleylek[1] ✉, Vincent Rijmen[2], Muharrem Tolga Sakallı[3], Emir Öztürk[3]

[1]*Department of Computer Engineering, Ondokuz Mayıs University, Samsun, Turkey*
[2]*Department of ESAT/COSIC, KU Leuven and iMinds, Leuven, Belgium*
[3]*Department of Computer Engineering, Trakya University, Edirne, Turkey*
✉ *E-mail: akleylek@gmail.com*

Abstract: In this study, the authors propose new methods using a divide-and-conquer strategy to generate $n \times n$ binary matrices (for composite n) with a high/maximum branch number and the same Hamming weight in each row and column. They introduce new types of binary matrices: namely, $(\boldsymbol{BHwC})_{t,\,m}$ and $(\boldsymbol{BCwC})_{q,\,m}$ types, which are a combination of Hadamard and circulant matrices, and the recursive use of circulant matrices, respectively. With the help of these hybrid structures, the search space to generate a binary matrix with a high/maximum branch number is drastically reduced. By using the proposed methods, they focus on generating 12×12, 16×16 and 32×32 binary matrices with a maximum or maximum achievable branch number and the lowest implementation costs (to the best of their knowledge) to be used in block ciphers. Then, they discuss the implementation properties of binary matrices generated and present experimental results for binary matrices in these sizes. Finally, they apply the proposed methods to larger sizes, i.e. 48×48, 64×64 and 80×80 binary matrices having some applications in secure multi-party computation and fully homomorphic encryption.

1 Introduction

Most block ciphers are composed of rounds where each round consists of a non-linear layer, a diffusion layer (including mixing layer such as MixColumns in Advanced Encryption Standard (AES) [1] and transposition layer such as ShiftRows in AES) and a key mixing layer. The non-linear layer (e.g. S-boxes) and the diffusion layer are mostly used elements in block cipher design which help to resist well-known attacks and are required to achieve Shannon's principles [2], which are confusion and diffusion, respectively. Feistel ciphers and substitution permutation networks (SPNs) with mixing layer are the most used structures in the design of block ciphers. An SPN has mainly two parts: a substitution layer and a diffusion layer (e.g. linear transformation). In a block cipher, a linear transformation maps a fixed sized input to the same size output by mixing the bits of the input to provide the required diffusion. It ensures that all output bits depend on all input bits after only a few rounds. In [3], it is shown that the linear transformation has a significant role in providing resistance against linear cryptanalysis [4] and differential cryptanalysis [5].

Two existing criteria to measure diffusion in linear transformations are the branch number [1] and the number of fixed points [6, 7]. The branch number helps us to define the security against linear and differential cryptanalysis and is a measure of the minimum number of active S-boxes for any two consecutive rounds in a block cipher.

Maximum distance separable (MDS) and maximum distance binary linear (MDBL) matrices (which mean $n \times n$ binary matrices with the maximum branch number which is equal to the maximum of the minimal distance of binary linear $[2n, n]$ codes) are used as diffusion layers in the round function of many block ciphers. AES, ANUBIS [8] and Khazad [9] are the examples of block ciphers using MDS matrices in their round function. Camellia [10], ARIA [11] and Pride [12] are the examples of block ciphers using MDBL matrices in their round function. Diffusion layers also called linear transformations are represented by matrices over $GF(2^s)$ or $GF(2)$. In the implementation phase, binary matrices have an advantage over MDS matrices since they need only exclusive or (XOR) operations to be implemented while MDS matrices may need XOR operations and scalings, which are implemented by table look-ups or rotations [13]. Recently, in [14], ciphers for secure multi-party computation (MPC) and fully homomorphic encryption (FHE) with low complexity were proposed. The sizes of binary linear layer of the proposed ciphers were relatively high, thus those were generated randomly. In this paper, we focus on binary linear transformations with a maximum/high branch number and good implementation properties.

1.1 Previous studies

In [15], a 32×32 binary matrix with branch number 10 (which is not the maximum achievable branch number for this size) is constructed with good implementation properties on 8, 32 and 64 bit platforms, which is aimed to transform a 256 bit input to a 256 bit output. In [16, 17], an algebraic method is presented to construct 8×8, 16×16 and 32×32 binary matrices with a maximum or maximum achievable branch number by taking into consideration the number of fixed points at the same time. These studies focus on achieving good implementation properties. Nevertheless, it seems that their results are not close to the optimal implementation properties (especially on 8 bit platforms) for 32×32 binary matrices. In [18], Sakallı *et al.* provide a method using a state transform matrix to generate $n \times n$ (where n is not a power of 2) binary matrices with a high branch number and a low number of fixed points. In [12], Albrecht *et al.* suggest using left circulant matrices with a maximum branch number. They focus on maximising the diffusion and minimising the density of the matrix for efficient implementation by using exhaustive search.

1.2 Motivation

The diffusion layer of block ciphers is getting more attention since there is a big demand to the cryptographic schemes running in resource constrained environments. Non-linear components of block ciphers such as S-boxes have been well studied. However, the studies on diffusion layers especially for binary linear transformations are very limited. Moreover, finding more constructions of linear transformations with good implementation properties is an open problem [12]. In [14], efficient constructions of binary linear layers with a high/maximum branch number and a

low XOR complexity for large sizes, to be used in environments when XORing is not for free, are stated as an open problem. In this paper, we focus on generating $n \times n$ binary matrices (for composite n) with a high/maximum branch number and a low implementation cost to be used in block ciphers. We estimate the implementation cost of a diffusion layer by the Hamming weight of the corresponding binary matrix. Note that the actual implementation cost might be lower.

1.3 Our contributions

Our main contribution is to present novel methods using the divide-and-conquer idea in order to generate $n \times n$ binary matrices with a high/maximum branch number and a minimum Hamming weight. Some attractive features of the proposed methods are:

i. The proposed methods are based on hybrid structures; i.e., Hadamard matrix with circulant matrices [$(\boldsymbol{BHwC})_{t,m}$] and circulant matrix with circulant matrices [$(\boldsymbol{BCwC})_{q,m}$].
ii. The search space to find an $n \times n$ binary matrix with a high/maximum branch number and a low implementation cost (low Hamming weight of rows/columns) is reduced from $\binom{n}{b}^n$ to $\binom{n}{b}$, where b is the odd Hamming weight of any row or column of $n \times n$ binary matrix.
iii. The proposed methods enable us to generate $n \times n$ binary diffusion layers having the same implementation properties as their inverse.
iv. By using $(\boldsymbol{BCwC})_{q,m}$ type (where $q = 3$, $m = 4$ or $q = 4$, $m = 3$), 12×12 binary matrices with the maximum branch number 8, which form, at the same time, the extended binary Golay codes ([24,12,8]), are generated.
v. Using the proposed methods, we generate 16×16 involutory (and orthogonal) and non-involutory binary matrices with the maximum branch number 8, and also give two selected examples with implementation properties better than the one in ARIA block cipher [11] (especially on 32 bit platforms).
vi. We apply $(\boldsymbol{BHwC})_{t,m}$ type (where $t = 2$ and $m = 8$) to generate 32×32 binary matrices with the maximum achievable branch number 12 and Hamming weight 11 in each row and column, and give some examples of binary matrices with good implementation properties (especially on 8 bit platforms). The experimental results show that the proposed method is very efficient since it generates 32×32 binary matrices (with the maximum achievable branch number 12) with a probability of 9.65% after searching through $C(32, 11) \simeq 2^{26.94}$ binary matrices.
vii. We provide some examples to emphasise the effectiveness of the proposed methods for $n \times n$ binary matrices with a high/maximum achievable branch number where $n \in \{48, 64, 80\}$.
viii. The experimental results show that the proposed methods are much more efficient than simple circulant approach to generate 16×16 binary matrices with the maximum branch number 8.
ix. The proposed methods can be used not only to generate binary matrices with a high/maximum branch number (an even value), but, also to generate binary matrices with a lower branch number (an even value), which maybe preferred in designing lightweight primitives (for example, Prince-like block ciphers).

1.4 Organisation

This paper is organised as follows: in Section 2, some basic definitions and their relevant properties are given and new binary diffusion layer structures are proposed. In Section 3, the proposed methods are explained and also some examples are provided to show that one can obtain $n \times n$ binary matrices with the desired properties. In Section 4, implementation details of the generated matrices with the proposed methods are discussed and compared with the previous ones. In Section 5, some interesting and nice experimental results of the proposed methods are provided and compared with simple circulant approach used in [12]. The conclusion is given in Section 6.

2 Preliminaries

In this section, we recall some basic definitions needed for generating binary diffusion layers. We start with the definitions of the special type matrices such as circulant and Hadamard matrices widely used in previous studies. Then, we define a new matrix type in Section 2.1 as a combination of Hadamard and circulant matrices [namely, $(\boldsymbol{BHwC})_{t,m}$] considered to be the main idea of the first proposed method. In Section 2.2, we use the idea of recursive use of circulant matrices to define the second matrix type [namely, $(\boldsymbol{BCwC})_{q,m}$]. Remark that all generated matrices in this paper are in the $(\boldsymbol{BHwC})_{t,m}$ or $(\boldsymbol{BCwC})_{q,m}$ type. Throughout this paper, q, m and t are positive integers.

Definition 1: An $m \times m$ circulant binary matrix is as follows:

$$\mathbf{circ}(a_1, a_2, \ldots, a_m) = \begin{bmatrix} a_1 & a_2 & \ldots & a_{m-1} & a_m \\ a_m & a_1 & \ldots & a_{m-2} & a_{m-1} \\ a_{m-1} & a_m & \ldots & a_{m-3} & a_{m-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_2 & a_3 & \ldots & a_m & a_1 \end{bmatrix}.$$

Remark 1: Circulant matrices over a binary field define a commutative algebra since $AB = BA$ and AB is also circulant. A circulant matrix $\boldsymbol{A}$ can be converted to its transpose, i.e. $\boldsymbol{A}^{\mathrm{T}}$ by a row or column permutation.

Definition 2 [8]: A $2^n \times 2^n$ binary Hadamard (**had**) matrix is in the following form:

$$\mathbf{had}(\boldsymbol{A}, \boldsymbol{B}) = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B} & \boldsymbol{A} \end{bmatrix}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are $2^{n-1} \times 2^{n-1}$ binary Hadamard matrices.

A $2^t \cdot m \times 2^t \cdot m$ block matrix $\boldsymbol{A}$ (with $2^t \times 2^t$ blocks of dimension $m \times m$) is called a Hadamard block matrix and denoted had($a_0, a_1, \ldots, a_{2^t-1}$) if and only if the blocks satisfy $\boldsymbol{A}_{i,j} = a_{i \oplus j}$ where $\boldsymbol{a}_i$s are $m \times m$ matrices, not necessarily in Hadamard form, for $0 \le i, j \le 2^t - 1$. Now, we recall the cryptographic properties of binary linear transformations.

Definition 3 [1]: Let $\boldsymbol{A}$ be an $n \times n$ binary matrix. Then, the differential branch number of $\boldsymbol{A}$ is defined as $\beta_d(\boldsymbol{A}) = \min\{wt(\boldsymbol{x}) + wt(\boldsymbol{A} \cdot \boldsymbol{x}^{\mathrm{T}}) | \boldsymbol{x} \in (GF(2))^n - \{0\}\}$. Similarly, the linear branch number of $\boldsymbol{A}$ is $\beta_l(\boldsymbol{A}) = \min\{wt(\boldsymbol{x}) + wt(\boldsymbol{A}^{\mathrm{T}} \cdot \boldsymbol{x}^{\mathrm{T}}) | \boldsymbol{x} \in (GF(2))^n - \{0\}\}$.

Note that the branch number of $\boldsymbol{A}$ and its inverse ($\boldsymbol{A}^{-1}$) are always the same, by definition (if $\boldsymbol{A}^{-1}$ exists).

Theorem 1 [19]: Let $\boldsymbol{A}$ be an $n \times n$ binary matrix and $\boldsymbol{I}$ be an $n \times n$ identity matrix. Let $\boldsymbol{G}_{n \times 2n} = [\boldsymbol{I}, \boldsymbol{A}]$ be the echelon form of a generating matrix of binary linear $[2n, n, d]$ code. Then, the branch number of $\boldsymbol{A}^{\mathrm{T}}$ is d.

In this paper, binary matrices with the same differential and linear branch number are generated by Lemmas 4 and 6. By Theorem 1, $\beta_{\max}$, the maximum branch number of an $n \times n$ binary matrix, is equal to the maximum of the minimal distance of binary linear $[2n, n]$ codes. In addition, the maximum branch number of an $n \times n$ binary matrix (for $n \le 18$) is known exactly since the achievable bounds and theoretical bounds are the same for these sizes. However, the achievable bounds and theoretical bounds may differ for $n > 18$ [19]. Throughout this paper, we use notion the maximum achievable branch number (referring to achievable bound) to denote the maximum branch number known of an $n \times n$ binary matrix where $n > 18$.

Definition 4 [20]: Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be binary matrices. $\boldsymbol{A}$ and $\boldsymbol{B}$ are called permutation homomorphic if there exists a row permutation σ and a column permutation γ with $\sigma(\gamma(\boldsymbol{A})) = \gamma(\sigma(\boldsymbol{A})) = \boldsymbol{B}$.

Lemma 1 [20]: If any two binary matrices are permutation homomorphic, then they have the same branch number.

Another cryptographic measure for a linear transformation (diffusion layer) is the number of fixed points [6, 7]. A fixed point is the point left intact after performing a diffusion layer $\boldsymbol{A}$, i.e. $\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{x}$ for $\boldsymbol{x} \in (GF(2^s))^n$. This means that there is no diffusion at fixed points since the input block at these points is the same after performing the diffusion layer.

Consider an S-box with output block with s-bit values (in practice generally, $s = 4$ or $s = 8$) to a diffusion layer $\boldsymbol{A}$. Let $\boldsymbol{A}$ be an $n \times n$ matrix over $GF(2^s)$, where $s \geq 1$ and $\boldsymbol{I}$ be an $n \times n$ identity matrix. Then, the solution of the following equation gives the set of all fixed points:

$$(\boldsymbol{A} + \boldsymbol{I}) \cdot \boldsymbol{x}^{\mathrm{T}} = 0,$$

where $\boldsymbol{x} \in (GF(2^s))^n$. The number of fixed points for a diffusion layer $\boldsymbol{A}$ is

$$F_A = 2^{s(n - \mathrm{rank}(\boldsymbol{A} + \boldsymbol{I}))}.$$

Note that if the matrix $(\boldsymbol{A} + \boldsymbol{I})$ has bigger rank, then the diffusion layer $\boldsymbol{A}$ has less fixed points. In Lemma 2, we give an interesting observation of the rank of $(\boldsymbol{A} + \boldsymbol{I})$ when $\boldsymbol{A}$ is involutory (self-inverse).

Lemma 2: Let $\boldsymbol{A}$ be an $n \times n$ binary involutory ($\boldsymbol{A} = \boldsymbol{A}^{-1}$) matrix and $\boldsymbol{I}$ be the $n \times n$ identity matrix. Then the rank of $(\boldsymbol{A} + \boldsymbol{I})$ is at most $n/2$.

Proof: Since $\boldsymbol{A}$ is involutory, we have $\boldsymbol{A}^2 = \boldsymbol{I}$. Then $\boldsymbol{A}^2 - \boldsymbol{I} = (\boldsymbol{A} - \boldsymbol{I})(\boldsymbol{A} + \boldsymbol{I}) = 0$ and $(\boldsymbol{A} + \boldsymbol{I})^2 = 0$ since we are working on $GF(2)$. $(\boldsymbol{A} + \boldsymbol{I})$ is 2-nilpotent matrix, i.e. row-reduced echelon form of $(\boldsymbol{A} + \boldsymbol{I})$ is upper triangular matrix with 0's in its diagonal. Recall that $\mathrm{Ker}(\boldsymbol{A} + \boldsymbol{I}) = \{\boldsymbol{x} \in (GF(2))^n | (\boldsymbol{A} + \boldsymbol{I}) \cdot \boldsymbol{x}^{\mathrm{T}} = 0\}$, $\dim(\mathrm{Ker}(\boldsymbol{A} + \boldsymbol{I})) + \mathrm{rank}(\boldsymbol{A} + \boldsymbol{I}) = n$ and $\mathrm{rank}(\boldsymbol{A} + \boldsymbol{I}) \leq \dim(\mathrm{Ker}(\boldsymbol{A} + \boldsymbol{I}))$. Then, rank of the matrix $(\boldsymbol{A} + \boldsymbol{I})$ is at most $n/2$.□

2.1 Binary Hadamard with circulant matrices

In this section, we give the main idea of the first proposed method (Section 3) for generating $n \times n$ binary linear transformations (where $n = 2^t \cdot m$) with a maximum branch number, which combines Hadamard and circulant matrices.

Definition 5: A binary Hadamard with circulant matrices, $(\boldsymbol{BHwC})_{t,m}$, is defined as a $(2^t \cdot m) \times (2^t \cdot m)$ binary matrix composed of $m \times m$ circulant matrices in a $2^t \times 2^t$ Hadamard block matrix.

Example 1: Let $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$ be 4×4 circulant binary matrices. Then $(\boldsymbol{BHwC})_{2,4}$ is a 16×16 binary matrix constructed as a $2^2 \times 2^2$ Hadamard block matrix with 4×4 circulant submatrices

$$(\boldsymbol{BHwC})_{2,4} = \mathbf{had}(\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3) = \begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_3 \\ \boldsymbol{A}_1 & \boldsymbol{A}_0 & \boldsymbol{A}_3 & \boldsymbol{A}_2 \\ \boldsymbol{A}_2 & \boldsymbol{A}_3 & \boldsymbol{A}_0 & \boldsymbol{A}_1 \\ \boldsymbol{A}_3 & \boldsymbol{A}_2 & \boldsymbol{A}_1 & \boldsymbol{A}_0 \end{bmatrix}_{16 \times 16}.$$

Lemma 3 shows the criterion of having involutory $(\boldsymbol{BHwC})_{t,m}$.

Lemma 3: Let $(\boldsymbol{BHwC})_{t,m}$ be a $(2^t \cdot m) \times (2^t \cdot m)$ binary matrix as in Definition 5. $(\boldsymbol{BHwC})_{t,m}$ is involutory, i.e. $(\boldsymbol{BHwC})_{t,m}^2 = \boldsymbol{I}$ if and only if $\sum_{i=0}^{2^t-1} \boldsymbol{A}_i$ is an involutory matrix.

Proof: The proof is similar to Lemma 1 in [16]. By definition

$$(\boldsymbol{BHwC})_{t,m} = \begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{A}_1 & \dots & \boldsymbol{A}_{2^t-2} & \boldsymbol{A}_{2^t-1} \\ \boldsymbol{A}_1 & \boldsymbol{A}_0 & \dots & \boldsymbol{A}_{2^t-1} & \boldsymbol{A}_{2^t-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{A}_{2^t-2} & \boldsymbol{A}_{2^t-1} & \dots & \boldsymbol{A}_0 & \boldsymbol{A}_1 \\ \boldsymbol{A}_{2^t-1} & \boldsymbol{A}_{2^t-2} & \dots & \boldsymbol{A}_1 & \boldsymbol{A}_0 \end{bmatrix}_{n \times n}$$

where for $0 \leq i \leq 2^t - 1$ $\boldsymbol{A}_i$s are $m \times m$ circulant matrices. Then

$$(\boldsymbol{BHwC})_{t,m}^2$$

$$= \begin{bmatrix} \sum_{i=0}^{2^t-1} \boldsymbol{A}_i^2 & \boldsymbol{B}_1 & \dots & \boldsymbol{B}_{2^t-2} & \boldsymbol{B}_{2^t-1} \\ \boldsymbol{B}_1 & \sum_{i=0}^{2^t-1} \boldsymbol{A}_i^2 & \dots & \boldsymbol{B}_{2^t-1} & \boldsymbol{B}_{2^t-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{B}_{2^t-2} & \boldsymbol{B}_{2^t-1} & \dots & \sum_{i=0}^{2^t-1} \boldsymbol{A}_i^2 & \boldsymbol{B}_1 \\ \boldsymbol{B}_{2^t-1} & \boldsymbol{B}_{2^t-2} & \dots & \boldsymbol{B}_1 & \sum_{i=0}^{2^t-1} \boldsymbol{A}_i^2 \end{bmatrix}_{n \times n} = \boldsymbol{I}$$

where for $1 \leq i \leq 2^t - 1$ $\boldsymbol{B}_i$s are $m \times m$ matrices. Each $\boldsymbol{B}_i$ consists of the multiplications $\boldsymbol{A}_j\boldsymbol{A}_k$ and $\boldsymbol{A}_k\boldsymbol{A}_j$ for some integers j and k due to the Hadamard property. Since circulant matrices form a commutative algebra ($\boldsymbol{A}_j\boldsymbol{A}_k = \boldsymbol{A}_k\boldsymbol{A}_j$) and we are working in $GF(2)$ ($\boldsymbol{A}_j\boldsymbol{A}_k + \boldsymbol{A}_k\boldsymbol{A}_j = 0$), $\boldsymbol{B}_i = 0$ for $1 \leq i \leq 2^t - 1$. By definition of the involutory matrix, in each row $\sum_{i=0}^{2^t-1} \boldsymbol{A}_i^2$ is an identity matrix. Note that $\sum_{i=0}^{2^t-1} \boldsymbol{A}_i^2 = \boldsymbol{I}$ if and only if $\sum_{i=0}^{2^t-1} \boldsymbol{A}_i$ is an involutory matrix thanks to the properties of $GF(2)$.□

Lemma 4: Any $(\boldsymbol{BHwC})_{t,m} = \mathbf{had}(\boldsymbol{A}_0, \dots, \boldsymbol{A}_{2^t-1})$ has the same differential and linear branch number.

Proof: By definition, any $(\boldsymbol{BHwC})_{t,m}$ consists of 2^t circulant matrices, which are used in a $2^t \times 2^t$ Hadamard block matrix. Hadamard block matrices are block-symmetric. Note that circulant matrices and their transposes have the same matrix properties since those are equivalent under certain row or column permutations by Remark 1. Then, they have the same branch number (see Lemma 1). Since a $(\boldsymbol{BHwC})_{t,m}$ is Hadamard block matrix, i.e. the matrix and its transpose are equal, any $(\boldsymbol{BHwC})_{t,m}$ has the same differential and linear branch number by definition of branch number. Recall that $wt(\boldsymbol{A} \cdot \boldsymbol{x}^{\mathrm{T}}) = wt(\boldsymbol{A}^{\mathrm{T}} \cdot \boldsymbol{x}^{\mathrm{T}})$ for $(\boldsymbol{BHwC})_{t,m}$ type.□

2.2 Binary circulant with circulant matrices

In this section, we give the idea of other proposed method, which use circulant matrices recursively (Section 3.2). Note that this matrix type can be used for generating $n \times n$ binary matrices with a maximum branch number, where $n > 1$. Recall that $(\boldsymbol{BHwC})_{t,m}$ can only be used for even dimensions such that $n = 2^t \cdot m$.

Definition 6: Let q and m be positive integers. A binary circulant with circulant matrices, $(\boldsymbol{BCwC})_{q,m}$, is defined as a $(q \cdot m) \times (q \cdot m)$ binary matrix constructed as a $q \times q$ circulant block matrix with $m \times m$ circulant submatrices.

Example 2: By using $(\boldsymbol{BCwC})_{q,m}$ type, one can construct 12×12 binary matrices in different ways. In this example, we give the two of them as follows:

- Let $\boldsymbol{A}_0$, $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ be 4×4 circulant binary matrices. Then $(\boldsymbol{BCwC})_{3,4}$ is a 12×12 binary matrix constructed as a 3×3 circulant block matrix with 4×4 circulant submatrices

$$(\boldsymbol{BCwC})_{3,4} = \mathbf{circ}(\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{A}_2) = \begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{A}_1 & \boldsymbol{A}_2 \\ \boldsymbol{A}_2 & \boldsymbol{A}_0 & \boldsymbol{A}_1 \\ \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_0 \end{bmatrix}_{12 \times 12}.$$

- Let $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$ be 3×3 circulant binary matrices. Then $(\boldsymbol{BCwC})_{4,3}$ is a 12×12 binary matrix constructed as a 4×4 circulant block matrix with 3×3 circulant submatrices

$$(\boldsymbol{BCwC})_{4,3} = \mathbf{circ}(\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3) = \begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_3 \\ \boldsymbol{A}_3 & \boldsymbol{A}_0 & \boldsymbol{A}_1 & \boldsymbol{A}_2 \\ \boldsymbol{A}_2 & \boldsymbol{A}_3 & \boldsymbol{A}_0 & \boldsymbol{A}_1 \\ \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_3 & \boldsymbol{A}_0 \end{bmatrix}_{12 \times 12}.$$

Remark 2: In this paper, we focus on the right circulant matrices. One can also extend Definition 6 to left circulant and mixed (right and left) circulant matrices. If n is prime (or q is chosen as 1), then Definition 6 is the same as Definition 1.

Lemma 5 shows the criterion of having involutory $(\boldsymbol{BCwC})_{q,m}$.

Lemma 5: Let $(\boldsymbol{BCwC})_{q,m}$ be a $(q \cdot m) \times (q \cdot m)$ binary matrix as in Definition 6. Let $\boldsymbol{A}_i$s be $m \times m$ circulant binary matrices and $\boldsymbol{I}$ be $m \times m$ identity matrix:

i. Let q be odd. $(\boldsymbol{BCwC})_{q,m}$ is involutory if and only if $\boldsymbol{A}_0^2 = \boldsymbol{I}$ ($\boldsymbol{A}_0$ is involutory) and $\boldsymbol{A}_i^2 = 0$ ($\boldsymbol{A}_i$ is 2-nilpotent matrix) for $1 \leq i < q$.
ii. Let q be even. $(\boldsymbol{BCwC})_{q,m}$ is involutory if and only if $(\boldsymbol{A}_0 + \boldsymbol{A}_{q/2})^2 = \boldsymbol{I}$ [$(\boldsymbol{A}_0 + \boldsymbol{A}_{q/2})$ is involutory] and $\boldsymbol{A}_i^2 + \boldsymbol{A}_{(q/2)+i}^2 = 0$ [$(\boldsymbol{A}_i + \boldsymbol{A}_{(q/2)+i})$ is 2-nilpotent matrix] for $1 \leq i < (q/2)$.

Proof: By definition

$$(\boldsymbol{BCwC})_{q,m} = \begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{A}_1 & \cdots & \boldsymbol{A}_{q-2} & \boldsymbol{A}_{q-1} \\ \boldsymbol{A}_{q-1} & \boldsymbol{A}_0 & \cdots & \boldsymbol{A}_{q-3} & \boldsymbol{A}_{q-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{A}_2 & \boldsymbol{A}_3 & \cdots & \boldsymbol{A}_0 & \boldsymbol{A}_1 \\ \boldsymbol{A}_1 & \boldsymbol{A}_2 & \cdots & \boldsymbol{A}_{q-1} & \boldsymbol{A}_0 \end{bmatrix}_{n \times n}$$

where $\boldsymbol{A}_i$s are $m \times m$ circulant matrices for $0 \leq i \leq q-1$.

i. Let q be odd. Then

$$(\boldsymbol{BCwC})_{q,m}^2 = \begin{bmatrix} \boldsymbol{A}_0^2 & \boldsymbol{A}_{(q+1)/2}^2 & \cdots & \boldsymbol{A}_{q-1}^2 & \boldsymbol{A}_{(q-1)/2}^2 \\ \boldsymbol{A}_{(q-1)/2}^2 & \boldsymbol{A}_0^2 & \cdots & \boldsymbol{A}_{(q-3)/2}^2 & \boldsymbol{A}_{q-1}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{A}_1^2 & \boldsymbol{A}_{q-2}^2 & \cdots & \boldsymbol{A}_0^2 & \boldsymbol{A}_{(q+1)/2}^2 \\ \boldsymbol{A}_{(q+1)/2}^2 & \boldsymbol{A}_1^2 & \cdots & \boldsymbol{A}_{(q-1)/2}^2 & \boldsymbol{A}_0^2 \end{bmatrix}_{n \times n}$$

$$= \boldsymbol{I}.$$

Since $(\boldsymbol{BCwC})_{q,m}^2$ is also a circulant matrix (see Remark 1), $\boldsymbol{A}_0^2 = \boldsymbol{I}$ and $\boldsymbol{A}_i^2 = 0$ for $1 \leq i < q$.

ii. Let q be even. Then

(see equation below)

Note that $(\boldsymbol{BCwC})_{q,m}^2$ is a circulant matrix. We have $\boldsymbol{A}_0^2 + \boldsymbol{A}_{q/2}^2 = \boldsymbol{I}$ and $(\boldsymbol{A}_0 + \boldsymbol{A}_{q/2})^2 = \boldsymbol{I}$ since we are working in $GF(2)$. Thus, $\boldsymbol{A}_0 + \boldsymbol{A}_{q/2}$ should be involutory. We obtain $(\boldsymbol{A}_{(q-2)/2}^2 + \boldsymbol{A}_{q-1}^2) = 0$, $\boldsymbol{A}_1^2 + \boldsymbol{A}_{(q+2)/2}^2 = 0$ and so on. In general form, $\boldsymbol{A}_i^2 + \boldsymbol{A}_{(q/2)+i}^2 = 0$ for $1 \leq i < (q/2)$. This results $(\boldsymbol{A}_i + \boldsymbol{A}_{(q/2)+i})^2 = 0$. Then, $(\boldsymbol{A}_i + \boldsymbol{A}_{(q/2)+i})$ should be nilpotent matrix of degree 2 for $1 \leq i \leq q-1$.□

Lemma 6: Any $(\boldsymbol{BCwC})_{q,m}$ has the same differential and linear branch number.

Proof: By Definition 1 and explanations below Definition 1, a circulant matrix can be converted to its transpose by a row or column permutation. By Lemma 1, if any two binary matrices are obtained by applying a row or column permutation of each other, they have the same branch number.□

3 Proposed methods for generating matrices

In this section, we present new methods using the divide-and-conquer idea to generate $n \times n$ binary matrices with a minimum Hamming weight, which have a maximum branch number. To the best of our knowledge, one can generate 12×12, 16×16 and 32×32 binary matrices with a maximum or maximum achievable branch number and the lowest implementation costs by using the proposed methods. Remark that the proposed methods are generic, i.e. one can search for binary matrices with a high even branch number for any dimension. In this paper, by combining Hadamard and circulant matrix properties, we focus on generating 12×12, 16×16 binary matrices with the maximum branch numbers 8 and Hamming weight 7 in all rows and columns and 32×32 binary matrices with the maximum achievable branch number 12 and Hamming weight 11 in all rows and columns, which have good implementation properties (on 8, 32 and 64 bit platforms).

The proposed methods are based on heuristic search, which drastically reduces the search space and generates binary matrices (with a maximum branch number) with a high probability. The experimental results (see Section 5) show that the diffusion layer

$$(\boldsymbol{BCwC})_{q,m}^2 = \begin{bmatrix} \boldsymbol{A}_0^2 + \boldsymbol{A}_{q/2}^2 & \boldsymbol{0} & \cdots & \boldsymbol{A}_{(q-2)/2}^2 + \boldsymbol{A}_{q-1}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_0^2 + \boldsymbol{A}_{q/2}^2 & \cdots & \boldsymbol{0} & \boldsymbol{A}_{(q-2)/2}^2 + \boldsymbol{A}_{q-1}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{A}_1^2 + \boldsymbol{A}_{(q+2)/2}^2 & \boldsymbol{0} & \cdots & \boldsymbol{A}_0^2 + \boldsymbol{A}_{q/2}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_1^2 + \boldsymbol{A}_{(q+2)/2}^2 & \cdots & \boldsymbol{0} & \boldsymbol{A}_0^2 + \boldsymbol{A}_{q/2}^2 \end{bmatrix}_{n \times n} = \boldsymbol{I}.$$

can be constructed in a reasonable time, faster than the previous techniques with a high success rate (SR). The procedure to generate $n \times n$ binary matrices with a maximum branch number consists of three main steps:

1. Finding suitable vectors whose Hamming weights sum is odd.
2. Constructing circulant matrices by using the vectors.
3. Building the $n \times n$ binary matrix in Hadamard or circulant matrix form with these circulant matrices [$(\boldsymbol{BHwC})_{t,m}$ or $(\boldsymbol{BCwC})_{q,m}$, see Definitions 5 and 6].

In short, the main idea behind the proposed methods for generating binary matrices with a maximum branch number is the nested structure (Hadamard block matrix with circulant submatrices or circulant block matrix with circulant submatrices). A Hadamard matrix (or circulant matrix) provides a stretching process on circulant matrices (satisfying high branch numbers) and allows at the same time to construct involutory binary matrices easily (see Lemmas 3 and 5).

3.1 Proposed method: $(\boldsymbol{BHwC})_{t,m}$ case

In this section, we follow Definition 5. Let $n = 2^t \cdot m$ where t and m are positive integers. The algorithm of the proposed method can be described as follows:

Step 1: Choose 2^t binary vectors for a fixed t where $t \in \{1, 2, 3, \ldots, (k-1)\}$ of length $n/2^t = m$ of the desired Hamming weight (see Remark 3). For example, for a 32×32 binary matrix, 2^2 binary vectors whose length is 8 are chosen. Observe that the sum of Hamming weight of 2^t binary vectors should be odd. Note that these are the first rows of the each circulant matrices used in step 2.

Step 2: Form $(n/2^t) \times (n/2^t)$ $(m \times m)$ circulant matrices from each vector.

Step 3: Construct $n \times n$ binary Hadamard matrix consisting of $2^t \times 2^t$ blocks [$(\boldsymbol{BHwC})_{t,m}$ type].

Step 4: Repeat the process for all possible n-bit values with a certain Hamming weight to generate all possible $n \times n$ matrices.

Step 5: Search for the $n \times n$ binary matrices (which are non-singular) with a maximum branch number.

Remark 3: The desired Hamming weight of 2^t binary vectors depends on the sum of Hamming weight of these vectors, which is obtained according to the required branch number of an $n \times n$ binary matrix and represents the Hamming weight of one row or column of an $n \times n$ binary matrix. The method uses odd Hamming weights.

By Remark 3 and $(\boldsymbol{BHwC})_{t,m}$ type, we focus on generating 16×16 (Section 3.1.1) and 32×32 (Section 3.1.2) binary matrices with branch numbers 8 and 12, respectively, and the minimum Hamming weights (which are 7 and 11 in all rows and columns, respectively) in order to have good implementation properties on software and hardware. Note also that the combination of Hadamard and circulant matrices in this method provide the same differential and linear branch number value for any $n \times n$ binary matrix (see Lemma 4).

Proposition 1: Let $wt(\boldsymbol{y})$ be an odd number and the Hamming weight of any row or column of an $n \times n$ binary matrix. Any binary matrix searched by the proposed method has both linear and differential branch number at most $wt(\boldsymbol{y}) + 1$, which is an even value.

Proof: By Definition 3, $(\min\{wt(\boldsymbol{x}) + wt(\boldsymbol{A} \cdot \boldsymbol{x}^{\mathrm{T}}) | \boldsymbol{x} \in (GF(2))^n - \{0\}\})$. Let the Hamming weight of any input x be 1. Let us consider a binary matrix for which $wt(\boldsymbol{y})$ is an odd Hamming weight of each row and column. Then, the maximum branch number of this binary matrix is at most $wt(\boldsymbol{y}) + 1$.□

Remark 4: If the sum of Hamming weight of 2^t binary vectors is chosen odd and $<n/2$, then binary matrices with good implementation properties can be generated as discussed in Sections 3.1.2 and 4.

3.1.1 Generation of 16×16 binary matrices with the maximum branch number 8 and good implementation properties by $(\boldsymbol{BHwC})_{2,4}$ type:

In this section, we present an example (Example 3) using $(\boldsymbol{BHwC})_{t,m}$ type, where $t = 2$ and $m = 4$. In Example 3, a 16×16 involutory and orthogonal ($\boldsymbol{A} \cdot \boldsymbol{A}^{\mathrm{T}} = \boldsymbol{I}$) binary matrix with branch number 8 is generated. Let a, b, c, d be the Hamming weight of 4 bit vectors. For example, for a 16 bit vector (0001–0011–0111–1111), $a = 1$, $b = 2$, $c = 3$, $d = 4$ and these vectors can be represented as (1–2–3–4). Then, the probable 16 bit values with four vectors of Hamming weight 7 are: (0–0–3–4), (0–1–2–4), (0–1–3–3), (0–2–2–3), (1–1–1–4), (1–2–2–2), (1–1–2–3). Note that in this representation the permutations of the vectors are not taken into consideration.

Example 3: Let $\boldsymbol{A} = (1000)$, $\boldsymbol{B} = (1000)$, $\boldsymbol{C} = (1000)$ and $\boldsymbol{D} = (1111)$ be the binary vectors. Note that (1000100010001111) in binary notation or (8–8–8–*F*) in hexadecimal notation is a 16 bit value of Hamming weight 7. This means that we can generate four circulant matrices of dimension 4×4. By applying step 2 of the proposed method, the four 4×4 circulant matrices ($\boldsymbol{A}_{\mathrm{circ}}$, $\boldsymbol{B}_{\mathrm{circ}}$, $\boldsymbol{C}_{\mathrm{circ}}$ and $\boldsymbol{D}_{\mathrm{circ}}$) can be obtained as follows:

$$\boldsymbol{A}_{\mathrm{circ}} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad \boldsymbol{B}_{\mathrm{circ}} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix},$$

$$\boldsymbol{C}_{\mathrm{circ}} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad \boldsymbol{D}_{\mathrm{circ}} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix}.$$

By applying step 3 of the proposed method, one can generate 16×16 involutory binary matrix with branch number 8 (with the combination of 4×4 Hadamard matrix) as given below

$$(\boldsymbol{BHwC})_{2,4} = \begin{bmatrix} \boldsymbol{A}_{\mathrm{circ}} & \boldsymbol{B}_{\mathrm{circ}} & \boldsymbol{C}_{\mathrm{circ}} & \boldsymbol{D}_{\mathrm{circ}} \\ \boldsymbol{B}_{\mathrm{circ}} & \boldsymbol{A}_{\mathrm{circ}} & \boldsymbol{D}_{\mathrm{circ}} & \boldsymbol{C}_{\mathrm{circ}} \\ \boldsymbol{C}_{\mathrm{circ}} & \boldsymbol{D}_{\mathrm{circ}} & \boldsymbol{A}_{\mathrm{circ}} & \boldsymbol{B}_{\mathrm{circ}} \\ \boldsymbol{D}_{\mathrm{circ}} & \boldsymbol{C}_{\mathrm{circ}} & \boldsymbol{B}_{\mathrm{circ}} & \boldsymbol{A}_{\mathrm{circ}} \end{bmatrix}$$

where each element of the 4×4 Hadamard matrix is 4×4 circulant matrix. The involutory (and orthogonal) binary matrix generated ($\boldsymbol{M}_1$) is

(see equation below)

Remark 5: One can construct a 16×16 involutory or non-involutory binary matrix with branch number 8 easily by using $(\boldsymbol{BHwC})_{2,4}$ type if the Hamming weights of three vectors from four vectors are chosen as 1 and the Hamming weight of the last vector is chosen as 4 with the vectors in any order (In Example 3, these vectors are denoted as $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$.).

3.1.2 Generation of 32×32 binary matrices with the maximum achievable branch number 12 and good implementation properties by $(\boldsymbol{BHwC})_{2,8}$ type:

In this section, we focus on generating 32×32 involutory and non-involutory (but having the same implementation cost for encryption and decryption) binary matrices with branch number 12 and Hamming weight 11 in all rows and columns. Then, we also show that these

generated matrices give better software and hardware implementation properties than the previous studies (see Table 2).

When generating 32×32 binary matrices with branch number 12, we use $(\boldsymbol{BHwC})_{2,8}$ type (4×4 Hadamard matrices are used, where each element of 4×4 Hadamard matrix is a 8×8 circulant matrix) with Hamming weight 11 in all rows and columns as follows:

$$(\boldsymbol{BHwC})_{2,8} = \mathbf{had}(\boldsymbol{A}_{\text{circ}}, \boldsymbol{B}_{\text{circ}}, \boldsymbol{C}_{\text{circ}}, \boldsymbol{D}_{\text{circ}}).$$

For a 32 bit value, $(a - b - c - d)$ means that Hamming weight of each vector (8 bit) is a, b, c and d, respectively. Then, the probable 32 bit values with four vectors of Hamming weight 11 are: (0–0–4–7), (0–0–3–8), 0–0–5–6), (0–1–2–8), (0–1–3–7), (0–1–4–6), (0–1–5–5), (0–2–2–7), (0–2–3–6), (0–2–4–5), (0–3–3–5), (0–3–4–4), (1–1–1–8), (1–1–2–7), (1–1–3–6), (1–1–4–5), (1–2–2–6), (1–2–3–5), (1–2–4–4), (1–3–3–4), (2–2–2–5), (2–2–3–4), (2–3–3–3).

The 32×32 binary matrices generated by using 32 bit values with four binary vectors (1–1–3–6) and (1–2–2–6) may have advantages to be implemented on 8 bit environments as shown in Section 4. In Examples 11 and 12 (provided in Appendix 1), the 32×32 binary matrices with branch number 12 and Hamming weight 11 in all rows and columns [(1–2–2–6) and (1–1–3–6)], respectively, are given. In addition to having the maximum achievable branch number 12, the binary matrices generated in the examples satisfy some criteria as follows:

- To have the same Hamming weight in each row and column (whether the binary matrices generated are involutory or not).
- To have efficient implementation properties on 8, 32 and 64 bit processors.
- To have a high rank of the matrix $(\boldsymbol{A}_{32\times 32} + \boldsymbol{I})$, where $\boldsymbol{I}$ is an identity matrix (it is 16 and 28 for involutory and non-involutory matrices, respectively, in this paper). Note that the generated 32×32 binary involutory matrices have the maximum achievable rank of the matrix $(\boldsymbol{A}_{32\times 32} + \boldsymbol{I})$ (see Lemma 2). For 32×32 non-involutory binary matrices, the highest achievable rank of the matrix $(\boldsymbol{A}_{32\times 32} + \boldsymbol{I})$ is 32. However, the maximum achievable rank of the matrix $(\boldsymbol{A}_{32\times 32} + \boldsymbol{I})$ for 32×32 binary matrices having odd value for the Hamming weight is 31.

Note that not all vectors are successful when generating 32×32 binary matrices with branch number 12. In Table 5, the number of binary matrices with branch number 12 for each successful vector is given. Any binary matrix with vectors includes permutations of blocks. We do not need to separately investigate permutations since we use $(\boldsymbol{BHwC})_{2,8}$ type and the permutations for any binary matrix provide different matrices (at most $2^t!$) of the same branch number by Lemma 7.

3.1.3 Generation of a 48 × 48 involutory binary matrix with the maximum achievable branch number 16 by using $(\boldsymbol{BHwC})_{3,6}$*:*

In this section, we present an example for a 48×48 involutory binary matrix with the maximum achievable branch number, i.e. 16 (Example 4).

Example 4: $(\boldsymbol{BHwC})_{3,6}$ is used to generate a 48×48 involutory binary matrix with branch number 16. Let $\boldsymbol{A} = (000001)$, $\boldsymbol{B} = (000010)$, $\boldsymbol{C} = (000101)$, $\boldsymbol{D} = (010001)$ and $\boldsymbol{E} = (111111)$ be the binary vectors. By applying step 2 of the proposed method, the five 6×6 circulant matrices ($\boldsymbol{A}_{\text{circ}}$, $\boldsymbol{B}_{\text{circ}}$, $\boldsymbol{C}_{\text{circ}}$, $\boldsymbol{D}_{\text{circ}}$ and $\boldsymbol{E}_{\text{circ}}$) are obtained. Then, according to step 3 of the proposed method, one can generate a 48×48 binary matrix with branch number 16 (with the combination of these 6×6 circulant matrices) as follows:

$$(\boldsymbol{BHwC})_{3,6} = \mathbf{had}(\boldsymbol{A}_{\text{circ}}, \boldsymbol{A}_{\text{circ}}, \boldsymbol{A}_{\text{circ}}, \boldsymbol{B}_{\text{circ}}, \boldsymbol{A}_{\text{circ}}, \boldsymbol{C}_{\text{circ}}, \boldsymbol{D}_{\text{circ}}, \boldsymbol{E}_{\text{circ}}),$$

where each element of the 8×8 Hadamard matrix is a 6×6 circulant matrix. Moreover, the Hamming weight of each row and column of the binary matrix is 15.

3.2 Proposed method: $(\boldsymbol{BCwC})_{q,m}$ *case*

In this section, we follow Definition 6. Let $n = q \cdot m$ where q and m are positive integers. By using the proposed method, an $n \times n$ binary matrix with an even branch number maybe generated since the method uses the benefits of odd Hamming weight in each row and column. The algorithm of the proposed method is very similar to the given one in Section 3.1:

Step 1: Choose $\boldsymbol{q}$ binary vectors where $\boldsymbol{q}$, $m \geq 1$ of length m with the desired Hamming weight (see Remark 3). For example, for a 12×12 binary matrix, three binary vectors whose length is 4 can be chosen or four binary vectors whose length is 3 can be chosen (see Example 2). Observe that the sum of Hamming weight of $\boldsymbol{q}$ binary vectors should be odd to obtain the maximum branch number. Note that these vectors are the first row of each circulant matrices used in step 2.

Step 2: Form $m \times m$ circulant matrices from each vector.

Step 3: Construct $n \times n$ binary matrix consisting of $q \times q$ blocks in circulant form [$(\boldsymbol{BCwC})_{q,m}$ type].

Step 4: Repeat the process for all possible n-bit values with a certain Hamming weight to generate all possible $n \times n$ matrices.

Step 5: Search for the $n \times n$ binary matrices (which are non-singular) with a maximum branch number.

$$\boldsymbol{M}_1 = \begin{bmatrix}
1&0&0&0&1&0&0&0&1&0&0&0&1&1&1&1\\
0&1&0&0&0&1&0&0&0&1&0&0&1&1&1&1\\
0&0&1&0&0&0&1&0&0&0&1&0&1&1&1&1\\
0&0&0&1&0&0&0&1&0&0&0&1&1&1&1&1\\
1&0&0&0&1&0&0&0&1&1&1&1&1&0&0&0\\
0&1&0&0&0&1&0&0&1&1&1&1&0&1&0&0\\
0&0&1&0&0&0&1&0&1&1&1&1&0&0&1&0\\
0&0&0&1&0&0&0&1&1&1&1&1&0&0&0&1\\
1&0&0&0&1&1&1&1&1&0&0&0&1&0&0&0\\
0&1&0&0&1&1&1&1&0&1&0&0&0&1&0&0\\
0&0&1&0&1&1&1&1&0&0&1&0&0&0&1&0\\
0&0&0&1&1&1&1&1&0&0&0&1&0&0&0&1\\
1&1&1&1&1&0&0&0&1&0&0&0&1&0&0&0\\
1&1&1&1&0&1&0&0&0&1&0&0&0&1&0&0\\
1&1&1&1&0&0&1&0&0&0&1&0&0&0&1&0\\
1&1&1&1&0&0&0&1&0&0&0&1&0&0&0&1
\end{bmatrix}.$$

3.2.1 Generation of 12 × 12 binary matrices with the maximum branch number 8 and good implementation properties by $(\boldsymbol{BCwC})_{3,4}$ and $(\boldsymbol{BCwC})_{4,3}$ types: In this section, we present two examples (Examples 5 and 6) by using $(\boldsymbol{BCwC})_{3,4}$ and $(\boldsymbol{BCwC})_{4,3}$ types denoted in Example 2 to generate 12 × 12 non-involutory binary matrices with branch number 8 (having the same implementation cost as their inverse).

Recall that the extended binary Golay code is known as [24, 12, 8] code with some desired properties, where a generator matrix is $\boldsymbol{G}_{12\times 24} = [\boldsymbol{I},\ \boldsymbol{C}]$ ($\boldsymbol{I}$ is the 12 × 12 identity matrix and $\boldsymbol{C}$ is a 12 × 12 binary matrix). The $\boldsymbol{G}_{12\times 24} = [\boldsymbol{I},\ \boldsymbol{M}_2]$ and $\boldsymbol{G}_{12\times 24} = [\boldsymbol{I},\ \boldsymbol{M}_3]$, where $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$ are the 12 × 12 non-involutory binary matrices given in Examples 5 and 6, respectively, are the extended binary Golay code since they satisfy the desired properties.

Example 5: Let $\boldsymbol{A} = (0011)$, $\boldsymbol{B} = (0011)$ and $\boldsymbol{C} = (1110)$ be the binary vectors. Note that (001100111110) in binary notation or (3–3–*E*) in hexadecimal notation is a 12 bit value of Hamming weight 7. One can generate a 12 × 12 non-involutory binary matrix with branch number 8 (with the combination of 3 × 3 circulant matrix) as $\boldsymbol{M}_2 = (\boldsymbol{BCwC})_{3,4} = \mathbf{circ}(\boldsymbol{A}_{\text{circ}}, \boldsymbol{B}_{\text{circ}}, \boldsymbol{C}_{\text{circ}})$, where each element of the 3 × 3 circulant matrix is a 4 × 4 circulant matrix. Note that the inverse of the 12 × 12 binary matrix $\boldsymbol{M}_2$ with the same implementation properties can be generated by using $(\boldsymbol{BCwC})_{3,4}$ type and the 12 bit value (011010110110) in binary notation or (6-B-6) in hexadecimal notation.

Example 6: Let $\boldsymbol{A} = (001)$, $\boldsymbol{B} = (001)$, $\boldsymbol{C} = (111)$ and $\boldsymbol{D} = (110)$ be the binary vectors. Note that (001001111110) in binary notation or (1–1–7–6) in hexadecimal notation is a 12 bit value of Hamming weight 7. One can generate a 12 × 12 non-involutory binary matrix with branch number 8 (with the combination of 4 × 4 circulant matrix) as $\boldsymbol{M}_3 = (\boldsymbol{BCwC})_{4,3} = \mathbf{circ}(\boldsymbol{A}_{\text{circ}}, \boldsymbol{B}_{\text{circ}}, \boldsymbol{C}_{\text{circ}}, \boldsymbol{D}_{\text{circ}})$, where each element of the 4 × 4 circulant matrix is a 3 × 3 circulant matrix. Note that the inverse of the 12 × 12 binary matrix $\boldsymbol{M}_3$ with the same implementation properties can be generated by using $(\boldsymbol{BCwC})_{4,3}$ type and the 12 bit value (010101111010) in binary notation or (2–5–7–2) in hexadecimal notation.

Remark 6: The rank of $(\boldsymbol{M}_2 + \boldsymbol{I})$ and $(\boldsymbol{M}_3 + \boldsymbol{I})$ matrices in Examples 5 and 6, respectively, is 11, which means that the binary matrices $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$ include two fixed points.

3.2.2 Generation of 16 × 16 binary matrices with the maximum branch number 8 and good implementation properties by $(\boldsymbol{BCwC})_{4,4}$ type: In this section, we present an example (Example 7) using $(\boldsymbol{BCwC})_{q,m}$ type, where $q = 4$ and $m = 4$. In Example 7, a 16 × 16 non-involutory binary matrix with branch number 8 (having the same implementation cost as its inverse) is generated. Note that the rank of $(\boldsymbol{M}_4 + \boldsymbol{I})$ matrix in Example 7 is 12.

Example 7: Let $\boldsymbol{A} = (0001)$, $\boldsymbol{B} = (0001)$, $\boldsymbol{C} = (0001)$ and $\boldsymbol{D} = (1111)$ be the binary vectors. Note that (0001000100011111) in binary notation or (1–1–1–*F*) in hexadecimal notation is a 16 bit value of Hamming weight of 7. By applying steps 2 and 3 of the proposed method, one can generate a 16 × 16 binary matrix with branch number 8 (with the combination of 4 × 4 circulant matrices) as $\boldsymbol{M}_4 = (\boldsymbol{BCwC})_{4,4} = \mathbf{circ}(\boldsymbol{A}_{\text{circ}}, \boldsymbol{B}_{\text{circ}}, \boldsymbol{C}_{\text{circ}}, \boldsymbol{D}_{\text{circ}})$, where each element of the 4 × 4 circulant matrix is a 4 × 4 circulant matrix. Note that the inverse of the 16 × 16 binary matrix $\boldsymbol{M}_4$ with the same implementation properties can be generated by using $(\boldsymbol{BCwC})_{4,4}$ type and the 16 bit value (0100111101000100) in binary notation or (4–*F*–4–4) in hexadecimal notation. In Appendix 1 (Example 10), we present an involutory 16 × 16 binary matrix with branch number 8 generated by $(\boldsymbol{BCwC})_{4,4}$ type and using (0010–0001–1111–0100) binary vectors.

3.2.3 Generation of 64 × 64 and 80 × 80 binary matrices with high branch numbers by using $(\boldsymbol{BCwC})_{8,8}$ and $(\boldsymbol{BCwC})_{10,8}$: In this section, we present examples for a 64 × 64 binary matrix with branch number 20 (Example 8) and an 80 × 80 binary matrix with branch number 22 (Example 9). For these sizes, the maximum achievable branch numbers are 22 and 23, respectively.

Example 8: $(\boldsymbol{BCwC})_{8,8}$ is used to generate a 64 × 64 binary matrix with branch number 20. Let $\boldsymbol{A} = (00000001)$, $\boldsymbol{B} = (00000010)$, $\boldsymbol{C} = (00000100)$, $\boldsymbol{D} = (00000011)$, $\boldsymbol{E} = (10001000)$, $\boldsymbol{F} = (00010100)$, $\boldsymbol{G} = (00001001)$ and $\boldsymbol{H} = (11111111)$ be binary vectors. By applying steps 2 and 3 of the proposed method, one can generate a 64 × 64 binary matrix with branch number 20 (with the combination of 8 × 8 circulant matrices) as follows:

$$(\boldsymbol{BCwC})_{8,8} = \mathbf{circ}(\boldsymbol{A}_{\text{circ}}, \boldsymbol{B}_{\text{circ}}, \boldsymbol{C}_{\text{circ}}, \boldsymbol{D}_{\text{circ}}, \boldsymbol{E}_{\text{circ}}, \boldsymbol{F}_{\text{circ}}, \boldsymbol{G}_{\text{circ}}, \boldsymbol{H}_{\text{circ}}),$$

where each element of the 8 × 8 circulant matrix is an 8 × 8 circulant matrix. Moreover, the Hamming weight of each row and column of the binary matrix is 19.

Example 9: $(\boldsymbol{BCwC})_{10,8}$ is used to generate an 80 × 80 binary matrix with branch number 22. Let $\boldsymbol{A} = (00000001)$, $\boldsymbol{B} = (00000011)$, $\boldsymbol{C} = (00000111)$, $\boldsymbol{D} = (00001111)$, $\boldsymbol{E} = (00011111)$, $\boldsymbol{F} = (00000100)$ and $\boldsymbol{G} = (10110010)$ be binary vectors. By applying steps 2 and 3 of the proposed method, one can generate an 80 × 80 binary matrix with branch number 22 (with the combination of 8 × 8 circulant matrices) as follows:

$$(\boldsymbol{BCwC})_{10,8} = \mathbf{circ}(\boldsymbol{A}_{\text{circ}}, \boldsymbol{A}_{\text{circ}}, \boldsymbol{A}_{\text{circ}}, \boldsymbol{B}_{\text{circ}}, \boldsymbol{C}_{\text{circ}}, \boldsymbol{D}_{\text{circ}}, \boldsymbol{E}_{\text{circ}}, \boldsymbol{A}_{\text{circ}}, \boldsymbol{F}_{\text{circ}}, \boldsymbol{G}_{\text{circ}}),$$

where each element of the 10 × 10 circulant matrix is a 8 × 8 circulant matrix. Moreover, the Hamming weight of each row and column of the binary matrix is 23.

4 Implementation details for the generated matrices

In this section, we give the implementation properties of the selected binary matrices of dimensions 12, 16 and 32. We focus on the implementation details of these matrices on 8, 32 and 64 bit platforms. We study the 16 × 16 and 32 × 32 binary matrices in detail since one can easily compare the efficiency of the generated matrices with similar matrices presented before. We also give the cryptographic properties of the generated matrices such as the number of fixed points and being involutory.

4.1 Implementation properties of the 12 × 12 binary matrices

The 12 × 12 binary matrices in Example 5 [$(\boldsymbol{BCwC})_{3,4}$] and Example 6 [$(\boldsymbol{BCwC})_{4,3}$] can be implemented on 8 bit platforms by 56 byte XORs and using four more temporary variables. On 32 bit platforms, they can be implemented by 15 word XORs (and using one more temporary variable) and 18 word XORs, respectively.

4.2 Implementation properties of the 16 × 16 binary matrices

In Table 1, the implementation properties on 8, 32 and 64 bit platforms of the 16 × 16 involutory binary matrix in Example 3 ($\boldsymbol{M}_1$) and non-involutory binary matrix in Example 7 ($\boldsymbol{M}_4$) are compared with the 16 × 16 involutory binary matrix used in the ARIA block cipher ($\boldsymbol{M}_{\text{ARIA}}$) [11]. The 16 × 16 binary matrices ($\boldsymbol{M}_1$ and $\boldsymbol{M}_4$) given in Examples 3 and 7 can be implemented by 52 XORs on 8 bit processors and using 12 temporary variables. Note that the implementation details of the binary matrix $\boldsymbol{M}_4$ on 8 bit processors are provided in Appendix 2. Moreover, they can be implemented by 15 word XORs and 9 double word XORs on 32 and 64 bit processors, respectively (and using only one temporary variable).

The diffusion layer of the block cipher ARIA is an involutory 16×16 binary matrix and includes 2^{72} fixed points because the rank of $(\boldsymbol{M}_{\text{ARIA}} + \boldsymbol{I})$ matrix is 7 and the inputs to the binary matrix are 16 8 bit S-box outputs. The binary matrix in Example 3 and Example 6 include 2^{64} and 2^{32} fixed points, respectively, when they are used with the same manner as in the ARIA. Note that the rank of $(\boldsymbol{M}_1 + \boldsymbol{I})$ matrix in Example 3 is 8, which is the maximum achievable rank for an involutory 16×16 binary matrix (see Lemma 2) and the rank of $(\boldsymbol{M}_4 + \boldsymbol{I})$ matrix in Example 7 is 12.

4.3 Implementation properties of the 32 × 32 binary matrices

In this paper, the 32×32 binary matrices are generated to have the minimum Hamming weight with respect to the branch number to be desired. Therefore, the matrices generated have very efficient implementation properties on 8 bit processors. They have also good implementation properties on 32 and 64 bit processors since the matrices generated are based on the combination of circulant and Hadamard matrices, i.e. some terms/patterns are repeated in other computations. In Table 2, 32×32 binary matrices with branch number 12 and branch number 10 found in the literature are compared with the matrices generated in this paper from viewpoint of their implementation properties on 8, 32 and 64 bit processors.

Now, we discuss implementation details of 32×32 binary matrices. Assume that we are dealing with one of the 32×32 binary matrices generated by the vectors (1–1–3–6) and (1–2–2–6). These binary matrices include a vector of Hamming weight 6 in their construction. Therefore, due to the Hadamard property, the 32 × 32 binary matrices generated with binary vectors (1–1–3–6) and (1–2–2–6) include the same four 8×8 circulant matrices. The singular 8×8 circulant binary matrix constructed by the vector **77** with Hamming weight 6 (represented by the value of the first row in hexadecimal notation) is as follows:

$$\mathbf{77}_{\text{circ}} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

Consider the equation $\boldsymbol{y} = \mathbf{77}_{\text{circ}} \cdot \boldsymbol{x}$ to be implemented on an 8 bit processor, where $\boldsymbol{x} = (x_0, x_1, \ldots, x_7)^{\text{T}}$, $\boldsymbol{y} = (y_0, y_1, \ldots, y_7)^{\text{T}}$ represent eight-dimensional binary input and output vectors, respectively. Also assume that x_i and y_i are byte values with $i \in \{0, 1, \ldots, 7\}$. A straightforward implementation needs 40 byte XORs. However, the total number of XOR operations can be reduced to 14 byte XORs by using six more temporary variables, which are

$$\begin{aligned} T_0 &= x_0 \oplus x_1 \oplus x_4 \oplus x_5, \\ T_1 &= x_2 \oplus x_3 \oplus x_6 \oplus x_7, \\ T_2 &= T_0 \oplus x_2 \oplus x_6, \\ T_3 &= T_0 \oplus x_3 \oplus x_7, \\ T_4 &= T_1 \oplus x_0 \oplus x_4, \\ T_5 &= T_1 \oplus x_1 \oplus x_5. \end{aligned}$$

The 32×32 binary matrices generated by the vectors (1–1–3–6) and (1–2–2–6) include the same four 8×8 circulant matrices of Hamming weight of 6 in their construction. That means 32×32 binary matrices can be implemented saving 104 byte XORs on 8 bit processors by using 24 more temporary variables if they include a vector of Hamming weight 6 with the same implementation property of the vector **77**. There are three more vectors providing this property, which are ***BB***, ***DD*** and ***EE***.

Remark 7: The vectors **77**, ***BB***, ***DD*** and ***EE*** provide 26 byte XORs saving on 8 bit processors as given above. However, there are eight vectors with Hamming weight 6, which can be used to construct 8 × 8 circulant binary matrices providing 22 byte XORs saving on 8 bit processors. These vectors are **5*F***, 7***D***, ***AF***, ***BE***, ***D*7**, ***EB***, ***F*5** and ***FA***. Note that they are represented by their first row in hexadecimal notation.

The 32×32 binary matrices with branch number 12 given in Examples 11 and 12 can be implemented by 216 and 232 byte XORs (using 24 more temporary variables), respectively.

Remark 8: One can also generate 32×32 binary matrices with branch number 10 by using the proposed methods. The Hamming weight of these matrices is 9 in all rows and columns [for example (1–1–1–6)]. It is possible to implement a 32×32 binary matrix having branch number 10 with 168 byte XORs instead of 256, 64 word XORs and 32 double word XORs on 8, 32 and 64 bit platforms, respectively. Note that the main improvement comes from the low Hamming weight (9 instead of 11) of each row.

To see the hardware performance of large binary matrices, i.e. 48×48, 64×64 and 80×80, Verilog hardware description language (HDL) code for all the matrices was written and synthesised by using *Synopsys DC Shell version H-2013.03-SP5-3* in the implementation process. To have a fair comparison, the implementations have been synthesised for three different technology libraries using typical operating conditions: 130 and 90 nm low-leakage Faraday libraries from United Microelectronics Corporation (UMC) and 45 nm generic NANGATE Open Cell Library. The resulting circuits were composed of only XOR gates

Table 1 Comparing 16 × 16 binary matrices given in the examples with that of ARIA from viewpoint of implementation properties on 8, 32 and 64 bit platforms

				Implementation cost		
	Branch number	Rank of $\boldsymbol{M} + \boldsymbol{I}$	Involutory	On 8 bit	On 32 bit	On 64 bit
$\boldsymbol{M}_{\text{ARIA}}$ [11]	8	7	yes	60 XORs	19 XORs	—
$\boldsymbol{M}_1$ in Example 3	8	8	yes	52 XORs	15 XORs	9 XORs
$\boldsymbol{M}_4$ in Example 7	8	12	no	52 XORs	15 XORs	9 XORs

Table 2 Comparing 32 × 32 binary matrices found in the literature from viewpoint of branch number and implementation properties on 8, 32 and 64 bit processors

				Implementation cost		
	Branch number	Rank of $\boldsymbol{A} + \boldsymbol{I}$	Involutory	On 8 bit	On 32 bit	On 64 bit
[15]	10	26	no	286 XORs	46 XORs	—
[17]	12	16	yes	328 XORs	—	—
$\boldsymbol{M}_6$ in Example 11	12	16	yes	216 XORs	80 XORs	40 XORs
$\boldsymbol{M}_7$ in Example 12	12	28	no	232 XORs	80 XORs	40 XORs

for all three libraries hence were identical for the same matrices. In Table 3, the required number of XOR operations for 48 × 48, 64 × 64 and 80 × 80 binary matrices after synthesising is given.

5 Experimental results

In this section, we discuss the efficiency of the proposed methods in view of the SR in generating binary matrices with a maximum branch number. We also provide the experimental results for simple circulant matrices (left circulant) denoted in [12] for comparison with the proposed methods since it is natural to ask whether the proposed methods give better results than the simple circulant approach. Note that MAGMA Computational Algebra Software and eight computers in parallel having Intel i7 central processing unit, 8 GB random access memory are used to get the results.

Table 3 Implementation results of 48 × 48, 64 × 64 and 80 × 80 binary matrices with maximum achievable/high branch numbers

$n \times n$	48 × 48	64 × 64	80 × 80
branch number	16	20	22
# XOR	316	597	931

Table 4 Experimental results for the proposed methods and circulant type with the selected sizes

n	Method	Search space	$\boldsymbol{M}$	SR, %
12	$(\boldsymbol{BCwC})_{3,4}$	$\binom{12}{7} = 792$	24	3.03
	$(\boldsymbol{BCwC})_{4,3}$		24	3.03
	circulant		24	3.03
16	$(\boldsymbol{BHwC})_{2,4}$	$\binom{16}{7} = 11,440$	4992	43.63
	$(\boldsymbol{BCwC})_{4,4}$		4736	41.40
	circulant		3136	27.41
32	$(\boldsymbol{BHwC})_{2,8}$	$\binom{32}{11} \simeq 2^{26.94}$	12,453,888	9.65

Table 5 Number of $n \times n$ binary matrices with maximum or maximum achievable branch number for each form

n	Method	Successful vectors	$\boldsymbol{bm}$	Total
12	$(\boldsymbol{BCwC})_{3,4}$	(2–2–3)	24	24
	$(\boldsymbol{BCwC})_{4,3}$	(1–1–2–3)	24	24
16	$(\boldsymbol{BHwC})_{2,4}$	(0–2–2–3)	1152	4992
		(1–1–1–4)	256	
		(1–2–2–2)	1280	
		(1–1–2–3)	2304	
	$(\boldsymbol{BCwC})_{4,4}$	(0–1–3–3)	512	4736
		(0–2–2–3)	768	
		(1–1–1–4)	128	
		(1–2–2–2)	1408	
		(1–1–2–3)	1920	
32	$(\boldsymbol{BHwC})_{2,8}$	(0–2–3–6)	61,440	12,453,888
		(0–2–4–5)	168,960	
		(0–3–3–5)	193,536	
		(0–3–4–4)	156,672	
		(1–1–3–6)	107,520	
		(1–1–4–5)	147,456	
		(1–2–2–6)	279,552	
		(1–2–3–5)	1,680,384	
		(1–2–4–4)	1,305,600	
		(1–3–3–4)	2,161,152	
		(2–2–2–5)	448,512	
		(2–2–3–4)	3,757,056	
		(2–3–3–3)	1,986,048	

Remark 9: By using the proposed method, the search space is drastically decreased. The experimental results show that binary linear transformations having good cryptographic properties can be efficiently found. In an ordinary search for $n \times n$ binary matrices with a certain Hamming weight, the search space is $\binom{n}{b}^n$, whereas the search space of the method proposed is $\binom{n}{b}$, where b denotes Hamming weight of n-bit value. By using different t values, the search space for the method maybe made larger for $(\boldsymbol{BHwC}_{t,m})$ type. On the other hand, the search space may also be made smaller because of the fact that $2^t \times 2^t$ Hadamard matrices are used in the construction (see Lemma 7). For $(\boldsymbol{BCwC})_{q,m}$ type, two null vectors of length m cannot appear in $\boldsymbol{q}$ vectors.

In Table 4, we present the SR of the methods by considering all search space for the given matrix types. In the experimental results, the 12 × 12 binary matrices with $(\boldsymbol{BCwC})_{3,4}$ and $(\boldsymbol{BCwC})_{4,3}$ types (see Section 3.2), the 16 × 16 binary matrices with $(\boldsymbol{BCwC})_{4,4}$ and $(\boldsymbol{BHwC})_{2,4}$ types and the 32 × 32 binary matrices with $(\boldsymbol{BHwC})_{2,8}$ (see Section 3.1) type are discussed. We also consider the Hamming weights of $n \times n$ binary matrices as 7, 7 and 11 in all rows and columns with $n = 12$, 16 and 32, respectively. For example, when searching 12 × 12 binary matrices with branch number 8, the search space is $\binom{12}{7} = 792$, where 7 is the Hamming weight of any row or column of a 12 × 12 binary matrix. In Table 4, n is the dimension of the binary matrix, $\boldsymbol{M}$ is the number of $n \times n$ binary matrices with a maximum or maximum achievable branch number and SR stands for the success rate of the method in generating binary matrices with a maximum or maximum achievable branch number.

The experimental results show that the proposed methods are much more efficient than the simple circulant method in generating $n \times n$ binary matrices with the maximum branch number for $n = 16$. The $(\boldsymbol{BCwC})_{q,m}$ or $(\boldsymbol{BHwC})_{t,m}$ type can be used to generate 16 × 16 or 32 × 32 binary matrices (with a maximum or maximum achievable branch number) with different parameters such that q, m and t satisfying $n = q \cdot m$ or $n = 2^t \cdot m$. For example, $(\boldsymbol{BCwC})_{4,8}$ can also be used when generating 32 × 32 binary matrices with branch number 12. Similarly, 16 × 16 binary matrices with the maximum branch number 8 can be generated by using $(\boldsymbol{BHwC})_{1,8}$ or $(\boldsymbol{BCwC})_{2,8}$ type.

Remark 10: The binary matrices generated with the proposed method and simple circulant matrices are different from each other. Moreover, they have advantages over simple circulant matrices from viewpoint of implementation on 8, 32 and 64 bit platforms (see Section 4).

In Table 5, we summarise that how many $n \times n$ for $n = 12$, 16 and 32 binary matrices with a maximum or maximum achievable branch number are generated by considering the set of successful vectors and using the proposed methods. Note that the successful vector is the representative of a vector group (which contains all permutations of the successful vector) enabling us to generate binary matrices with a maximum branch number. The third column represents the successful vectors that generate binary matrices with a maximum or maximum achievable branch number and contains the Hamming weights of the first row of the binary matrix. In addition, $\boldsymbol{bm}$ is the number of $n \times n$ binary matrices with a maximum or maximum achievable branch numbers (8, 8 and 12 for 12 × 12, 16 × 16 and 32 × 32 binary matrices, respectively) generated by all permutations of successful vectors.

Remark 11: Each permutation of the successful vectors generates the same number of binary matrices with a maximum branch number if a binary matrix is in $(\boldsymbol{BHwC})_{t,m}$ type. In some cases, this is not applicable for the binary matrices in $(\boldsymbol{BCwC})_{q,m}$ type.

By Remark 11, for example, any permutation of the successful vector with Hamming weight (1–1–1–4) generates 64 16 × 16 binary matrices with branch number 8 (Table 5) and the vector with

Hamming weight (1–1–1–4) totally generates 256 binary matrices with the same branch number since it has four different permutations. The 12×12 binary matrices [obtained by $(\boldsymbol{BCwC})_{3,4}$ type] with branch number 8 are generated by only the successful vector with Hamming weight (2–2–3). All permutations of this vector in terms of Hamming weight are (2–2–3), (2–3–2) and (3–2–2), generate the same number of binary matrix (which is 8) with the maximum branch number. On the other hand, the 12×12 binary matrices [obtained by $(\boldsymbol{BCwC})_{4,3}$ type] with branch number 8 are generated by only the successful vector with Hamming weight (1–1–2–3). This vector has 12 different permutations. From these permutations, eight of them are successful and these permutations in terms of Hamming weight are as follows: (1–1–2–3), (1–1–3–2), (1–2–3–1), (1–3–2–1), (2–1–1–3), (2–3–1–1), (3–1–1–2) and (3–2–1–1).

Lemma 7: By using the idea in $(\boldsymbol{BHwC})_{t,m}$ or $(\boldsymbol{BCwC})_{q,m}$ type, one can generate from each successful vector at most $2^t!$ or q! $n \times n$ different binary matrices of the same branch number, respectively. These $n \times n$ binary matrices are related to each other by row and column permutations. Moreover, they are permutation homomorphic.

Proof: By Remark 9, Lemmas 1 and 7, the proof is trivial.□

6 Conclusion and future work

In this paper, we present heuristic search algorithms to generate $n \times n$ binary matrices with a maximum branch number and a minimum Hamming weight by using two new binary matrix types: namely, $(\boldsymbol{BHwC})_{t,m}$ and $(\boldsymbol{BCwC})_{q,m}$. Moreover, the search space to find $n \times n$ binary matrices with a maximum branch number is significantly reduced by using a divide-and-conquer strategy. The $(\boldsymbol{BHwC})_{t,m}$ type has an advantage over the $(\boldsymbol{BCwC})_{q,m}$ type if one wants to generate an involutory binary matrix or a non-involutory binary matrix with the same implementation properties as its inverse. On the other hand, the $(\boldsymbol{BCwC})_{q,m}$ type has an advantage over the $(\boldsymbol{BHwC})_{t,m}$ type from the viewpoint of the number of different parameters in generating $n \times n$ binary matrices with a maximum branch number.

The experimental results (up to size 32) show that $n \times n$ binary matrices with a maximum branch number (which is even) and a minimum Hamming weight can be generated for most composite sizes $n \leq 32$ except for 24. The implementation results show that we generate 12×12, 16×16 and 32×32 binary matrices (with a maximum or maximum achievable branch number) with the lowest implementation costs. The proposed method is better than the previous studies in view of implementation properties in generating binary matrices having good cryptographic properties. Moreover, we propose a novel solution, superior to random search in terms of branch number, to construct binary diffusion layers with high branch number for $n > 32$ having some applications in block ciphers for MPC and FHE. Therefore, we would like to note that the proposed methods can be used efficiently for the design of lightweight block ciphers. We believe that our results shed an additional light on the generation of binary diffusion layers.

7 Acknowledgments

Sedat Akleylek is partially supported by TÜBITAK under 2219-Postdoctoral Research Program Grant. The authors would like to express their gratitude to the anonymous reviewers for their invaluable suggestions in putting the present study into its final form.

8 References

[1] Daemen, J., Rijmen, V.: '*The design of Rijndael, AES – the advanced encryption standard*' (Springer, Heidelberg, 2002)
[2] Shannon, C.E.: 'Communication theory of secrecy systems', *Bell Syst. Tech. J.*, 1949, **28**, (7), pp. 656–715
[3] Daemen, J.: 'Cipher and hash function design'. PhD thesis, Katholieke Universiteit Leuven, Belgium, 1995
[4] Matsui, M.: 'Linear cryptanalysis method for DES cipher'. Proc. of EUROCRYPT'93, 1994 (LNCS, **765**), pp. 386–397
[5] Biham, E., Shamir, A.: 'Differential cryptanalysis of DES-like cryptosystems'. Proc. of CRYPTO'90, 1990 (LNCS, **537**), pp. 2–21
[6] Flajolet, P., Sedgewick, R.: '*Analytic combinatorics*' (Cambridge University Press, Cambridge, 2009)
[7] Z'aba, M.R.: 'Analysis of linear relationships in block ciphers'. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2010
[8] Barreto, P.S.L.M., Rijmen, V.: 'The ANUBIS legacy-level block cipher'. Proc. First Open NESSIE Workshop, Leuven, 2000
[9] Barreto, P.S.L.M., Rijmen, V.: 'The Khazad legacy-level block cipher'. Proc. First Open NESSIE Workshop, Leuven, 2000
[10] Aoki, K., Ichikawa, T., Kanda, M., *et al.*: 'Camellia: a 128 bit block cipher suitable for multiple platforms-design and analysis'. Proc. of Selected Areas in Cryptography, SAC 2000, 2001 (LNCS, **2012**), pp. 39–56
[11] Kwon, D., Kim, J., Park, S., *et al.*: 'New block cipher: ARIA'. Proc. of Int. Conf. on Information Security and Cryptology, 2004 (LNCS, **2971**), pp. 432–445
[12] Albrecht, M.R., Driessen, B., Kavun, E.B., *et al.*: 'Block ciphers – focus on the linear layer (feat. PRIDE)'. CRYPTO (1), 2014 (LNCS, **8616**), pp. 57–76
[13] Nakahara, J.Jr., Abrahão, É.: 'A new involutory MDS matrix for the AES', *Int. J. Netw. Sec.*, 2009, **9**, (2), pp. 109–116
[14] Albrecht, M.R., Rechberger, C., Schneider, T., *et al.*: 'Ciphers for MPC and FHE'. Proc. of EUROCRYPT 2015 Part I, 2015 (LNCS, **9056**), pp. 430–454
[15] Koo, B.W., Jang, H.S., Song, J.H.: 'On constructing of a 32×32 binary matrix as a diffusion layer for a 256 bit block cipher'. Proc. of Int. Conf. on Information Security and Cryptology, 2006 (LNCS, **4296**), pp. 51–64
[16] Aslan, B., Sakallı, M.T.: 'Algebraic construction of cryptographically good binary linear transformations', *Sec. Commun. Netw.*, 2014, **7**, (1), pp. 53–63
[17] Sakallı, M.T., Aslan, B.: 'On the algebraic construction of cryptographically good 32×32 binary linear transformations', *J. Comput. Appl. Math.*, 2014, **259**, (Part B), pp. 485–494
[18] Sakallı, M.T., Akleylek, S., Aslan, B., *et al.*: 'On the construction of 20×20 and 24×24 binary matrices with good implementation properties for lightweight block ciphers and hash functions', *Math. Prob. Eng.*, doi: 10.1155/2014/540253
[19] Kwon, D., Sung, S.H., Song, J.H., *et al.*: 'Design of block ciphers and coding theory', *Trends Math.*, 2005, **8**, (1), pp. 13–20
[20] Koo, B.W., Jang, H.S., Song, J.H.: 'Constructing and cryptanalysis of a 16×16 binary matrix as a diffusion layer'. Proc. of Information Security Applications Fourth Int. Workshop (WISA 2003), 2003 (LNCS, **2908**), pp. 489–503

9 Appendix

9.1 Appendix 1: Examples

Example 10: Let 16 bit value of Hamming weight 7 be of four binary vectors (1–1–4–1). Then, choose (2–1–*F*–4) in hexadecimal notation or (0010–0001–1111–0100) in binary notation. Using the proposed method given in Section 3, $\boldsymbol{M}_5 = (\boldsymbol{BCwC})_{4,4}$ is the involutory 16×16 binary matrix with branch number 8.

Example 11: Let 32 bit value of Hamming weight 11 be of four binary vectors (1–2–2–6). Then, choose (1–3–82–77) in hexadecimal notation or (00000001–00000011–10000010–01110111) in binary notation. Using the proposed method given in Section 3, $\boldsymbol{M}_6 = (\boldsymbol{BHwC})_{2,8}$ is the involutory 32×32 binary matrix with branch number 12.

Example 12: Let 32 bit value of Hamming weight 11 be of four binary vectors (1–1–3–6). Then, choose (80–20–*C*8–7*D*) in hexadecimal notation or (10000000–00100000–11001000–01111101) in binary notation. Using the proposed method given in Section 3, $\boldsymbol{M}_7 = (\boldsymbol{BHwC})_{2,8}$ is the non-involutory 32×32 binary matrix with branch number 12. Note that the inverse of the 32×32 binary matrix $\boldsymbol{M}_6$ with the same implementation properties can be generated by using $(\boldsymbol{BHwC})_{2,8}$ type and the 32 bit value (2–80–23–*F*5) in hexadecimal notation.

9.2 Appendix 2: implementation

9.2.1 Implementation of the 16 × 16 binary matrix given in Example 7 on 8 bit processors: If the 16×16 binary matrix given in Example 7 is implemented on a 8 bit processor, then $\boldsymbol{M}_4$ is represented by byte XORs of binary vectors with $\boldsymbol{y} = \boldsymbol{M}_4 \cdot \boldsymbol{x}$, where $\boldsymbol{x} = (x_0, x_1, \ldots, x_{15})^{\mathrm{T}}$, $\boldsymbol{y} = (y_0, y_1, \ldots, y_{15})^{\mathrm{T}}$ with $x_i, y_i \in GF(2^8)$, $i = 0, 1, \ldots, 15$. Note also that T_0, T_1, T_2 and T_3 are temporary

variables used to reduce the number of operations to 60 XORs. Then

$$
\begin{aligned}
T_0 &= x_0 \oplus x_1 \oplus x_2 \oplus x_3,\\
T_1 &= x_4 \oplus x_5 \oplus x_6 \oplus x_7,\\
T_2 &= x_8 \oplus x_9 \oplus x_{10} \oplus x_{11},\\
T_3 &= x_{12} \oplus x_{13} \oplus x_{14} \oplus x_{15},\\
y_0 &= T_3 \oplus x_3 \oplus x_7 \oplus x_{11},\\
y_1 &= T_3 \oplus x_0 \oplus x_4 \oplus x_8,\\
y_2 &= T_3 \oplus x_1 \oplus x_5 \oplus x_9,\\
y_3 &= T_3 \oplus x_2 \oplus x_6 \oplus x_{10},\\
y_4 &= T_0 \oplus x_7 \oplus x_{11} \oplus x_{15},\\
y_5 &= T_0 \oplus x_4 \oplus x_8 \oplus x_{12},\\
y_6 &= T_0 \oplus x_5 \oplus x_9 \oplus x_{13},\\
y_7 &= T_0 \oplus x_6 \oplus x_{10} \oplus x_{14},\\
y_8 &= T_1 \oplus x_3 \oplus x_{11} \oplus x_{15},\\
y_9 &= T_1 \oplus x_0 \oplus x_8 \oplus x_{12},\\
y_{10} &= T_1 \oplus x_1 \oplus x_9 \oplus x_{13},\\
y_{11} &= T_1 \oplus x_2 \oplus x_{10} \oplus x_{14},\\
y_{12} &= T_2 \oplus x_3 \oplus x_7 \oplus x_{15},\\
y_{13} &= T_2 \oplus x_0 \oplus x_4 \oplus x_{12},\\
y_{14} &= T_2 \oplus x_1 \oplus x_5 \oplus x_{13},\\
y_{15} &= T_2 \oplus x_2 \oplus x_6 \oplus x_{14}.
\end{aligned}
$$

The number of XOR operations can be reduced to 52 by using 8 more temporary variables, which are: $T_4 = x_3 \oplus x_7$, $T_5 = x_0 \oplus x_4$, $T_6 = x_1 \oplus x_5$, $T_7 = x_2 \oplus x_6$, $T_8 = x_{11} \oplus x_{15}$, $T_9 = x_8 \oplus x_{12}$, $T_{10} = x_9 \oplus x_{13}$ and $T_{11} = x_{10} \oplus x_{14}$.