Defence Science Journal, Vol. 74, No. 1, January 2024, pp. 68-78, DOI : 10.14429/dsj.74.18824

Construction of New Hadamard Matrix Forms to Generate 4×4 and 8×8 Involutory MDS Matrices Over *GF*(2^m) for Lightweight Cryptography

Yogesh Kumar[#,*], P.R. Mishra[#], Atul Gaur[$] and Gaurav Mittal[!]

[#]*DRDO-Scientific Analysis Group, Delhi-110 054, India*
[$]*Department of Mathematics, University of Delhi, Delhi-110 007, India*
[!]*DRDO-Joint Cipher Bureau, Delhi-110 054, India*
[*]*E-mail: adhana.yogesh@gmail.com*

ABSTRACT

In this paper, we present the construction of two Hadamard matrix forms over $GF(2^m)$ to generate 4×4 and 8×8 involutory MDS (IMDS) matrices. The first form provides a straightforward way to generate 4×4 IMDS matrices, while the second is an efficient way to generate 8×8 IMDS matrices using a hybrid (combination of search-based methods and direct construction) approach. In addition, we propose an algorithm for computing the branch number of any non-singular matrix over $GF(2^m)$ and improve its computational complexity for Hadamard matrices. Using this algorithm and the proposed Hadamard matrix form, we obtain $2^k \times 2^k$ lightweight involutory and non-involutory Hadamard MDS matrices with low XOR counts for k=2,3. Finally, we carry out a comparative study based on the XOR count to demonstrate that MDS matrices created using our Hadamard matrix forms have lower XOR counts than MDS matrices available in the literature as of today.

Keywords: Finite field; Branch number; Diffusion; MDS matrices; Cryptography

1. INTRODUCTION

Confusion and diffusion, as defined by Claude Shannon[1], are the mandatory characteristics needed in the construction of secure cryptographic primitives such as block ciphers and hash functions. In general, substitution boxes (or S-boxes) and linear transformations are used to achieve these properties in block ciphers and hash functions. Furthermore, the linear transformation induced by an MDS (maximum distance separable) matrix is a popular choice as the core component of the diffusion layer since it provides optimal diffusion. There is a well-known concept of branch number[2]that can be used to measure the diffusion capabilities of a linear transformation. Since MDS matrices have the maximum possible branch number, they are particularly utilised to infuse security against well-recognized attacks, for example, differential and linear cryprtanalysis[3-4].

In this study, we are concerned about the generation of MDS matrices that are useful for lightweight cryptography (LC). In order to produce an LC design, it is essential that even the basic building blocks should have lightweight properties. This has a direct impact since the usage of such building blocks consumes a lesser number of the logical gates (XOR operations) in the implementation. Therefore, the cost of hardware implementation would be lower. Involutory MDS (IMDS) matrices are preferable choices in several block ciphers, hash functions, and stream ciphers due to the same execution cost in encryption/decryption phases and less area occupancy (see[5] for a comprehensive overview). Consequently, researchers around the world are exploring for novel techniques to build IMDS matrices for LC.

In general, MDS matrix construction can be divided into two parts:(i) direct construction and (ii) search-based construction. The direct construction is primarily based on Cauchy matrices[6-7], Vandermonde matrices[8], companion matrices[9-10] etc. However, it may be noted that Cauchy and Vandermonde matrices based constructions are not efficient for lower-cost implementations (see[11]). Consequently, the majority of the search-based constructions of MDS matrices comprise hybrid structures[12], recursive methods[13-14], heuristic approach[15], and searching the matrix forms like circulant and Hadamard matrix forms[16-17]. Moreover, several other matrix forms used to find MDS matrices for LC are Toeplitz, and Hankel matrices (see[11] for a nice overview). The search-based constructions of MDS matrices involve checking whether it is MDS or not for specific choices of its elements. However, these constructions are useful for matrices of small orders because checking whether a matrix is MDS or not, is still computationally expensive. Recently, the authors of[18] proposed a new approach to construct MDS matrices using generalised Cauchy matrices, and they found interesting connections between the entries of any arbitrary 3×3 sub-matrix of the constructed MDS matrix. In addition, they investigated MDS matrices with maximum possible 1's.

One of the aims in this paper is to minimize the amount of circuit area required to implement the MDS matrices for

Received : 10 April 2023, Revised : 05 September 2023
Accepted : 06 September 2023, Online published : 12 January 2024

LC (see[19] for deeper understanding). The XOR count is a measure used to examine the lightweightness of a matrix (see Definition 3). For MDS matrices, lightweightness is not an intrinsic property, but it depends on the irreducible polynomial (IP) chosen to generate the underlying field. Generally, the low Hamming weight field elements require fewer hardware resources to implement the multiplication in $GF(2^m)$ and therefore, it is taken as the common criterion for choosing a matrix. However, for some particular choice of the polynomial defining the finite field, even for a high Hamming weight element, the multiplication in $GF(2^m)$ may be implemented with a much lower XOR count[16].

Guzel[20], *et al.* devised a new matrix form for generating 3×3 IMDS matrices over a finite field. In continuation with the previous work, Pehlivanoglu[16], *et al.* presented a generalized Hadamard (GHadamard) form for generating MDS matrices for LC. Using their GHadamard form, Pehlivanoglu *et al.* get better results than the best-known results of XOR count for several $2^k\times2^k$ non-IMDS and IMDS matrices over $GF(2^m)$ where k=2,3. Recently, Yang[5], *et al.* proposed a computationally effective method to locate lightweight IMDS matrices for LC using the idea of global optimisation and improved various well-known results of XOR count. Our contribution to this paper is discussed in the next subsection.

We present the generation of $2^k\times2^k$ lightweight IMDS (LIMDS) matrices over the finite fields $GF(2^4)$ and $GF(2^8)$ for k=2,3. The rationale behind the selection of dimensions 4×4 and 8×8 is that most of the well-known ciphers and hash functions use matrices of these dimensions to induce diffusion[21-22]). To generate the LIMDS matrices, we suggest two new Hadamard matrix forms over $GF(2^m)$ and use them to find $2^k\times2^k$ IMDS matrix representatives for k=2,3. Further, the generalised Hadamard (GHadamard) matrix form can be used to produce IMDS matrices[10]. Using our matrix forms, we show that the number of 4×4 and 8×8 IMDS matrices over $GF(2^m)$ respectively are $r_1\times(2^m-1)^2$ and $r_2\times(2^m-1)^3$, where r_1 and r_2, respectively, represent the number of all 4×4 and 8×8 IMDS matrix representatives.

We explicitly compute the number of IMDS matrices of order 4×4 over $GF(2^8)$ whereas in the case of 8×8 involutory MDS matrices, we provide an algorithm for computing the same (see Algorithm 2). Algorithm 2 requires the computation of branch number (see section 2 for its formal definition) of non-singular matrices. We provide an algorithm (see Algorithm 1) for computing the branch number of any non-singular matrix over $GF(2^m)$. We explicitly deduce the computational complexity of Algorithm 1 and improve the computational complexity in the case of Hadamard matrix form over $GF(2^m)$. Further, we also generate $2^k\times2^k$ (k=2,3) lightweight non-IMDS Hadamard matrices by exhaustively computing the branch number through Algorithm 1 with the smallest XOR counts. Finally, we present a comparative study based on XOR count to show that our constructed MDS matrices have low XOR counts in comparison to the known XOR counts of MDS matrices in the literature.

This paper is organised as follows. In the next section, we describe mathematical preliminaries including MDS matrices and XOR count required to implement the multiplication in $GF(2^m)$. In section 3, we propose a direct construction of 4×4 IMDS matrices and a hybrid construction, i.e., a combination of search-based methods and direct construction, of 8×8 IMDS matrices over $GF(2^m)$. We discuss the strategy for computing the branch number of any non-singular matrix over $GF(2^m)$ in the same section. Furthermore, in section 3, we also generate 4×4 and 8×8 lightweight non-IMDS Hadamard matrices with lesser XOR counts than the known XOR counts. We exhibit our experimental results in section 4 and present a comparative study based on the XOR count. Finally, we conclude the paper in section 5.

2. MATHEMATICAL PRELIMINARIES

This section discusses certain definitions and the mathematical preliminaries needed to comprehend the paper. We construct the finite field $GF(2^m)$ from the prime field $GF(2)$ as a residue class ring $GF(2)[x]/(f(x))$, where, $f(x)$ is an irreducible polynomial in $GF(2)[x]$ of degree m. The residue class ring $GF(2)[x]/(f(x))$, consists of residue classes $g+(f(x))$ denoted by $[g]$ with $g \in GF(2)[x]$. Two residue classes $[g]$ and $[h]$ are identical precisely if g-h is divisible by f. Each residue class contains a unique representative $r\in GF(2)[x]$ with deg(r)<deg(f), which is simply the remainder in the division of g by f. The distinct residue classes are exactly $r+(f(x))$ where, r runs through all polynomials in $GF(2)[x]$ with deg(r)<deg(f). Then the cardinality of $GF(2)[x]/(f(x))$, is the count of total polynomials in $GF(2)[x]$ of degree $<m$, which is precisely 2^m. The necessary background of finite fields can be found in[23].

The differential branch number[24] $B_d(M)$ of a matrix M of order n over $GF(2^m)$ is defined as:

$$B_d(M) = \min_{x\neq 0}\{w_h(x) + w_h(M.x)\},$$

where, $w_h(x)$ denotes the weight of the vector x. Also, the linear branch number[24] $B_l(M)$ of a matrix M over $GF(2^m)$ is defined as:

$$B_l(M) = \min_{x\neq 0}\{w_h(x) + w_h(M^T.x)\}\cdot$$

Our focus in this work is on the Hadamard matrix form, which is symmetric. For this matrix form, differential and linear branch numbers are the same. We write $B_d(M)$ simply as the branch number and denote it as $B(M)$.

Definition 1. A $2^t\times2^t$ Hadamard matrix H over $GF(2^m)$ for $t > 0$ is defined as:

$$H = had(A_0, A_1) = \begin{bmatrix} A_0 & A_1 \\ A_1 & A_0 \end{bmatrix},$$

where, sub matrices A_0 and A_1 are also $2^{t-1}\times2^{t-1}$ Hadamard matrices. Some results of Hadamard matrix H over $GF(2^m)$ are as follows[10]:

$H_{i,j}$ = $a_{i\oplus j}$. Here, a_i's are first-row elements of a Hadamard matrix H.

H^2 = $c^2\times I$ where, $c = \oplus_{i=0}^{k-1} a_i$ and I is the $2^t\times2^t$ identity matrix. If c=1, then H is an involutory matrix.

Definition 2.[16] Let $H=had(a_0,a_1,\ldots,a_{k-1})$ be a $k\times k$ Hadamard matrix over $GF(2^m)$ with $k=2^t$ for t >0. Then a $k\times k$ GHadamard matrix $GH=(a_0,a_1;b_1,a_2;b_2,\ldots,a_{k-1};\ b_{k-1})$ can be expressed as follows:

$$GH = \begin{bmatrix} a_0 & a_1 b_1 & \cdots & a_{k\text{-}2} b_{k\text{-}2} & a_{k\text{-}1} b_{k\text{-}1} \\ a_1 b_1^{-1} & a_0 & \cdots & a_{k\text{-}1} b_1^{-1} b_{k\text{-}2} & a_{k\text{-}2} b_1^{-1} b_{k\text{-}1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{k\text{-}2} b_{k-2}^{-1} & a_{k\text{-}1} b_{k-2}^{-1} b_1 & \cdots & a_0 & a_1 b_{k-2}^{-1} b_{k\text{-}1} \\ a_{k\text{-}1} b_{k-1}^{-1} & a_{k\text{-}2} b_{k-1}^{-1} b_1 & \cdots & a_1 b_{k-1}^{-1} b_{k\text{-}2} & a_0 \end{bmatrix}$$

Let $GH_{i,j}$ be the $(i,j)^{th}$ entry GH. Then, $GH_{i,j} = a_{i\oplus j} b_i^{-1} b_j$, where, $0 \leq i \leq k\text{-}1$; $1 \leq j \leq k\text{-}1$, b_i 's are non-zero element of $GF(2^m)$ and $b_0 = b_0^{-1} = 1$.

Next, in Table 1, we give the abbreviations used in this paper.

Table 1. Abbreviations

Abbreviation	Meaning
MDS	Maximum distance separable
LC	Lightweight cryptography
LIMDS	Lightweight involutory MDS matrices
IMDS	Involutory MDS
Non-IMDS	Non-involutory MDS
IP(s)	Irreducible polynomial(s)
CC	Computational complexity

2.1 Analysis of XOR Count

Definition 3. $XOR(a)$ is the minimum XOR operations needed to compute the multiplication of $a \in GF(2^m)$ with any $b \in GF(2^m)$.

The distribution of XOR counts in a finite field is not an intrinsic property of the finite field. It is rather dependent on the underlying generating polynomial used for implementing field multiplication. Given m≥2 the total XOR count i.e., $\sum_{a\in GF(2^m)} XOR(a)$ is independent of the generating polynomial and is equal to $m\sum_{i=2}^{m} 2^{i\text{-}2}(i\text{ - }1)$, where, m≥2, (See[12]). It means different XOR count distributions in a finite field with respect to different generating polynomials have the same mean but different variances. If the variance of some generating polynomials is too low, the XOR counts will predominantly lie near the mean, and the XOR count of an arbitrary n×n IMDS matrix will be nearly constant for such a distribution. Such a polynomial is not very useful for searching lightweight IMDS matrices. As a result, we choose polynomials with high variance values to find a low XOR count matrix. We choose the polynomials $0x13[x^4+x+1]$ and $0x1c3[x^8+x^7+x^6+x+1]$ for $GF(2^4)$ and $GF(2^8)$. The variances of these polynomials are 57.7490 and 56.7490, respectively.

Let us begin with a straightforward approach for calculating the number of XOR operations required to perform the product of $a \in GF(2)[x]/(0x1c3)$ with an element x over $GF(2)[x]/(0x1c3)$. Let $a=(a_7,a_6,\ldots,a_0)$ be in hexadecimal coefficient form, then the corresponding polynomial form in $GF(2)[x]/(0x1c3)$ is $a_7x^7+a_6x^6+\ldots+a_0$. We see that:

$$\begin{aligned} a.x &= (a_7x^7 + a_6x^6 + \ldots + a_0).x \\ &= (a_7x^8 + a_6x^7 + a_5x^6 + a_4x^5 + a_3x^4 + a_2x^3 + a_1x^2 + a_0x) \\ &= a_7(x^7 + x^6 + x + 1) + a_6x^7 + a_5x^6 + a_4x^5 \\ &\quad + a_3x^4 + a_2x^3 + a_1x^2 + a_0x \\ &= (a_7 \oplus a_6, a_7 \oplus a_5,\ a_4, a_3,\ a_2, a_1,\ a_7 \oplus a_0, a_7). \end{aligned}$$

As a result, the XOR count in $a.x$ is three. In general, we can write

$$a.x = (msb(a) * 0 \times c3) \oplus (a << 1),$$

where, $msb(a)$ represents the most significant bit of a. Furthermore, when a is multiplied by x^2, we see that:

$$\begin{aligned} &a.x^2 \\ &= (a_7 \oplus a_6, a_7 \oplus a_5,\ a_4, a_3,\ a_2, a_1,\ a_7 \oplus a_0, a_7).x \\ &= (a_6 \oplus a_5, a_7 \oplus a_6 \oplus a_4,\ a_3,\ a_2, a_1,\ a_7 \oplus a_0, a_6, a_7 \oplus a_6). \end{aligned}$$

As a result, the XOR count for $a.x^2$ is 5. Furthermore, while multiplying a with x^3, we notice that:

$$\begin{aligned} &a.x^3 \\ &= (a_6 \oplus a_5, a_6 \oplus a_5, 0, 0, 0, 0, a_6 \oplus a_5, a_6 \oplus a_5) \\ &\oplus (a_7 \oplus a_6 \oplus a_4,\ a_3,\ a_2, a_1, a_7 \oplus a_0, a_6, a_7 \oplus a_6, 0) \\ &= (a_7 \oplus a_5 \oplus a_4, a_6 \oplus a_5 \oplus a_3,\ a_2, a_1, a_7 \oplus a_0, a_6, \\ &\quad a_7 \oplus a_5, a_6 \oplus a_5). \end{aligned}$$

The XOR count for $a.x^3$ is 7. Similarly, we can calculate the XOR counts for $a.x^4, a.x^5, a.x^6,$ and $a.x^7$. Furthermore, if $b=b_7x^7+b_6x^6+\ldots+b_0$ is any arbitrary element in $GF(2)[x]/(0x1c3)$, then we can write:

$$\begin{aligned} a.b &= a.(b_7x^7 + b_6x^6 + \ldots + b_0) \\ &= a.b_7x^7 + a.b_6x^6 + a.b_5x^5 + a.b_4x^4 + a.b_3x^3 + a.b_2x^2 \\ &\quad + a.b_1x^1 + a.b_0. \end{aligned}$$

We can then compute the number of XOR operations required to multiply $a \in GF(2)[x]/(0x1c3)$ with any element b.

Next, we compute the number of XOR operations required in implementing a row of an $n \times n$ matrix over $GF(2^m)$. Let $x_0, x_1, \ldots, x_{n\text{-}1}$ be the elements of any row of an $n \times n$ matrix over $GF(2^m)$. Let $\alpha_0, \alpha_1, \ldots \alpha_{n\text{-}1}$ be the XOR counts required to execute the multiplication of $x_0, x_1, \ldots, x_{n\text{-}1}$ with another field element. Then, the number of XOR operations required to implement a row is determined by $(\alpha_0 + \alpha_1 + \ldots + \alpha_{n\text{-}1}) + (w\text{ - }1).m$, where, m is the dimension of the finite filed $GF(2^m)$ over $GF(2)$ and w is the number of non-zero elements in the row. We also compute the number of XOR operations required in the implementation of a $n \times n$ matrix over $GF(2^m)$. Let $M=(m_{i,j})$ be an $n \times n$ matrix and $X=(x_1, x_2, \ldots, x_n)^T$ be a column vector in $GF(2^m)$. Then, $M.X = (\sum_{k=1}^{n} m_{1k}x_k, \sum_{k=1}^{n} m_{2k}x_k, \ldots, \sum_{k=1}^{n} m_{nk}x_k)$. Clearly, $M.X$ contains n entries. For each entry, n-field multiplications and $(n\text{-}1)$ additions are required. As a result, there will be n^2 field multiplications and $n(n\text{-}1)$ additions. In this case, n^2 field multiplications will require $\sum_{i=1}^{n}\sum_{j=1}^{n} XOR(m_{i,j})$ XOR operations and $n(n\text{-}1)$ additions will require $n(n\text{-}1).m$ XOR operations, where, m is the dimension of the finite field $GF(2^m)$ over $GF(2)$. The number of XOR operations required to implement a $n \times n$ matrix over $GF(2^m)$ now becomes:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} XOR(m_{i,j}) + n(n\text{ - }1)m.$$

We also develop a tool that computes the number of XOR operations required to implement the product of $a \in GF(2)[x]/(f(x))$ with an arbitrary element b over $GF(2^m)$ for $m=1,2,\ldots,16$ where, $f(x)$ is an IP of degree m. Using this tool, we present the amount of XOR operations required to implement the product of $a \in GF(2^8)$ with b over $GF(2^8)$ using two different polynomials $0x1c3$ and $0x11d$ in Table 5.

3. THE PROPOSED METHOD

We recall that there are two generic strategies in the literature for obtaining an MDS matrix. The first is direct construction, which ensures that the constructed matrix is MDS. The second is to use testing to filter MDS matrices from a given collection of matrices. The present paper comes into both of these categories. The proposed methods include a direct construction of 4×4 IMDS matrices over $GF(2^m)$, and a hybrid construction of 8×8 IMDS matrices over $GF(2^m)$ which is a combination of search-based methods and direct construction.

The key idea of the proposed methods is first to generate 4×4 and 8×8 IMDS matrix representatives and then to obtain $2^k \times 2^k$ IMDS matrices by applying some non-zero parameters along with their inverses to these IMDS matrix representatives for k=2,3. These parameters preserve MDS and involutory properties of any given 4×4 and 8×8 IMDS matrix representatives.

Apart from the ease of optimizing XOR counts, using involutory matrices provides an added advantage. All square submatrices in an MDS matrix should be non-singular. When we work with $n \times n$ involutory matrices over $GF(2^m)\backslash\{0\}$, we know that every $(n-1)\times(n-1)$ submatrix is invertible. As a result, one of the matrix's required conditions for being MDS is already met. To generate an involutory MDS matrix representative, we use generic properties of a Hadamard matrix that satisfy the involutory property, namely, the XOR sum of the elements in any row/column of a Hadamard matrix is 1 and the XOR sum of the elements in the main diagonal and anti-diagonal (counter diagonal) is equal to 0.

Plan of section: In subsections 3.1 and 3.2, we generate 4×4 and 8×8 IMDS matrices over $GF(2^8)$ for LC, respectively. In subsection 3.2, we also discuss an algorithm for computing the branch number of non-singular matrices and improve its complexity in the case of Hadamard matrices. In subsection 3.3, we present the generation of 4×4 and 8×8 lightweight non-IMDS Hadamard matrices by exhaustively computing the branch number.

3.1 GENERATION OF 4×4 IMDS MATRICES

We define the Hadamard matrix form M_1 in order to search for 4×4 IMDS matrix representatives over $GF(2^m)$ as follows:

$$M_1 = \begin{bmatrix} 1 & \beta_0 & \beta_1 & \beta_0+\beta_1 \\ \beta_0 & 1 & \beta_0+\beta_1 & \beta_1 \\ \beta_1 & \beta_0+\beta_1 & 1 & \beta_0 \\ \beta_0+\beta_1 & \beta_1 & \beta_0 & 1 \end{bmatrix}.$$

The Hadamard matrix form M_1 for deducing 4×4 IMDS matrix representatives is explicitly defined by two parameters β_0 and β_1 over $GF(2^m)\backslash\{0\}$. The search space for finding 4×4 IMDS matrix representatives is $(2^m-1)^2$. The following theorem states that if β_0 and β_1 are chosen in a specific way, the search over this space can be eliminated.

Theorem 1. The matrix M_1 defined above is an IMDS if β_0 and β_1 satisfy the following conditions:

1. $1,\beta_0,\beta_1$ are linearly independent
2. $\beta_0\beta_1^{-1} \neq \beta_0+\beta_1$
3. $\beta_1\beta_0^{-1} \neq \beta_0+\beta_1$
4. $\beta_0^{-1}+\beta_1^{-1} \neq 1$

Proof: M_1 is invertible because it is a Hadamard matrix with a sum of entries in each row and column of 1. Now we prove that the proposed form M_1 is an MDS matrix if the four conditions listed above are met. All square submatrices in an MDS matrix should be non-singular. Note that each entry in a 4×4 involutory matrix is its co-factor. This fact makes every 3×3 submatrix of M_1 non-singular as we choose our entries to be the non-zero elements of $GF(2^m)$. It is still necessary to demonstrate that the determinant of every 2×2 sub-matrix is not zero. If we choose $\beta_0,\beta_1 \in GF(2^m)$ such that $1,\beta_0,\beta_1$ are linearly independent over $GF(2)$ and satisfy the conditions laid down in (*b*), (*c*) and (*d*) i.e., $\beta_0 \neq \beta_1 \neq 1$, $\beta_0+\beta_1 \neq 1$ and $\beta_0\beta_1^{-1} \neq \beta_0+\beta_1, \beta_1\beta_0^{-1} \neq \beta_0+\beta_1, \beta_0^{-1}+\beta_1^{-1} \neq 1$. Then determinant of every 2×2 sub-matrix is non-zero. This completes the proof.

When conditions on β_0 and β_1 laid down in Theorem 1 were asserted on M_1, a total of 63,252 IMDS matrix representatives of order 4×4 were found over $GF(2^8)$.

Let $RIM=[m_{ij}]$ represent a 4×4 IMDS matrix representative. Then, the GHadamard matrix form GRIM obtained by the matrix RIM and some parameters b_i's for i=1,2,3 is also involutory MDS in the following form:

$$M = \begin{bmatrix} m_{11} & m_{12}b_1 & m_{13}b_2 & m_{14}b_3 \\ m_{21}b_1^{-1} & m_{22} & m_{23}b_1^{-1}b_2 & m_{24}b_1^{-1}b_3 \\ m_{31}b_2^{-1} & m_{32}b_2^{-1}b_1 & m_{33} & m_{34}b_2^{-1}b_3 \\ m_{41}b_3^{-1} & m_{42}b_3^{-1}b_1 & m_{43}b_3^{-1}b_2 & m_{44} \end{bmatrix},$$

where, b_i's for i=1,2,3 are the elements of $GF(2^m)\backslash\{0\}$. There are:

$63252\times(2^8-1)^3=1048805131500$

IMDS matrices of order 4×4 over $GF(2^8)$.

3.2 Generation of 8×8 IMDS Matrices

We define the Hadamard matrix form M_2 in order to search for 8×8 IMDS matrix representatives over $GF(2^m)$ (matrix 1).

The Hadamard matrix form M_2 for deducing 8×8 IMDS matrix representatives is explicitly defined by three parameters $(\beta_0,\beta_1,\beta_2)$ over $GF(2^m)\backslash\{0\}$. The search space for finding 8×8 IMDS matrix representatives is $(2^m-1)^3$.

Let $RIM=[m_{ij}]$ represent an 8×8 IMDS matrix representative. Then, the GHadamard matrix form GRIM derived by the matrix RIM and some parameters b_i's for i=1,2,…,7 is also IMDS in the Matrix 2 form.

The number of 8×8 IMDS matrices over $GF(2^m)$ is given by $r_2\times(2^m-1)^7$, where, r_2 represents the number of 8×8 IMDS matrix representatives. We are interested in finding the instances of M_2 where, it is MDS. Instead of verifying for non-vanishing minors, we employ the fact that an $n\times n$ matrix over $GF(2^m)$ is an MDS matrix if and only if its branch number is n+1. The original problem is now reduced to finding the instances of matrix M_1 with branch numbers of precisely nine.

We discuss the computational complexity (CC) of an algorithm for computing the branch number of non-singular matrices (based on observations of Guo[2], *et al.*). Furthermore, we improve the CC of this algorithm in the case of Hadamard matrices. This will be discussed in the subsections 3.2.1 and

$$\begin{bmatrix}
1 & \beta_0 & \beta_1 & \beta_1+1 & \beta_0+\beta_1+1 & \beta_2 & \beta_2+\beta_0 & \beta_1+\beta_0 \\
\beta_0 & 1 & \beta_1+1 & \beta_1 & \beta_2 & \beta_0+\beta_1+1 & \beta_1+\beta_0 & \beta_2+\beta_0 \\
\beta_1 & \beta_1+1 & 1 & \beta_0 & \beta_2+\beta_0 & \beta_1+\beta_0 & \beta_0+\beta_1+1 & \beta_2 \\
\beta_1+1 & \beta_1 & \beta_0 & 1 & \beta_1+\beta_0 & \beta_2+\beta_0 & \beta_2 & \beta_0+\beta_1+1 \\
\beta_0+\beta_1+1 & \beta_2 & \beta_2+\beta_0 & \beta_1+\beta_0 & 1 & \beta_0 & \beta_1 & \beta_1+1 \\
\beta_2 & \beta_0+\beta_1+1 & \beta_1+\beta_0 & \beta_2+\beta_0 & \beta_0 & 1 & \beta_1+1 & \beta_1 \\
\beta_2+\beta_0 & \beta_1+\beta_0 & \beta_0+\beta_1+1 & \beta_2 & \beta_1 & \beta_1+1 & 1 & \beta_0 \\
\beta_1+\beta_0 & \beta_2+\beta_0 & \beta_2 & \beta_0+\beta_1+1 & \beta_1+1 & \beta_1 & \beta_0 & 1
\end{bmatrix}$$

Matrix 1.

$$\begin{bmatrix}
m_{11} & m_{12}b_1 & m_{13}b_2 & m_{14}b_3 & m_{15}b_4 & m_{16}b_5 & m_{17}b_6 & m_{18}b_7 \\
m_{21}b_1^{-1} & m_{22} & m_{23}b_1^{-1}b_2 & m_{24}b_1^{-1}b_3 & m_{25}b_1^{-1}b_4 & m_{26}b_1^{-1}b_5 & m_{27}b_1^{-1}b_6 & m_{28}b_1^{-1}b_7 \\
m_{31}b_2^{-1} & m_{32}b_2^{-1}b_1 & m_{33} & m_{34}b_2^{-1}b_3 & m_{35}b_2^{-1}b_4 & m_{36}b_2^{-1}b_5 & m_{37}b_2^{-1}b_6 & m_{38}b_2^{-1}b_7 \\
m_{41}b_3^{-1} & m_{42}b_3^{-1}b_1 & m_{43}b_3^{-1}b_2 & m_{44} & m_{45}b_3^{-1}b_4 & m_{46}b_3^{-1}b_5 & m_{47}b_3^{-1}b_6 & m_{48}b_3^{-1}b_7 \\
m_{51}b_4^{-1} & m_{52}b_4^{-1}b_1 & m_{53}b_4^{-1}b_2 & m_{54}b_4^{-1}b_3 & m_{55} & m_{56}b_4^{-1}b_5 & m_{57}b_4^{-1}b_6 & m_{58}b_4^{-1}b_7 \\
m_{61}b_5^{-1} & m_{62}b_5^{-1}b_1 & m_{63}b_5^{-1}b_2 & m_{64}b_5^{-1}b_3 & m_{65}b_5^{-1}b_4 & m_{66} & m_{67}b_5^{-1}b_6 & m_{68}b_5^{-1}b_7 \\
m_{71}b_6^{-1} & m_{72}b_6^{-1}b_1 & m_{73}b_6^{-1}b_2 & m_{74}b_6^{-1}b_3 & m_{75}b_6^{-1}b_4 & m_{76}b_6^{-1}b_5 & m_{77} & m_{78}b_6^{-1}b_7 \\
m_{81}b_7^{-1} & m_{82}b_7^{-1}b_1 & m_{83}b_7^{-1}b_2 & m_{84}b_7^{-1}b_3 & m_{85}b_7^{-1}b_4 & m_{86}b_7^{-1}b_5 & m_{87}b_7^{-1}b_6 & m_{88}
\end{bmatrix}$$

Matrix 2.

3.2.2. At the end of subsection 3.2.2, we discuss an algorithm for computing r_2 and deduce its CC.

3.2.1 Algorithm for Computing Branch Number of Non-Singular Matrices

Guo[2], *et al.* observed that the branch number d of a binary $n\times n$ non-singular matrix M can be determined by searching for the minimum value of $w_h(x)+w_h(A.x)$, where, $A=M,M^{-1}$ among the input vectors of weight up to $d/2$, $d\leq n+1$. However, the algorithm proposed in[2] is only confined to binary matrices.

We generalize the idea of[2] to compute the branch number of any non-singular matrix M of order $n\times n$ over $GF(2^m)$. We define the set T_l, $1\leq l\leq n$ as the collection of column vectors of weight l given as:

$T_l = \{[a_0,a_1,...,a_{n-1}]^T \text{ and } a_0,a_1,...,a_{n-1}\in GF(2^m),\ wt([a_0,a_1,...,a_{n-1}])=l\}$.

To that goal, we develop the algorithm as follows:

Algorithm 1

Computation of branch number of $n\times n$ non-singular matrix *M over* $GF(2^m)$.

---

function GETBRANCHNUMBER(M,n)
$B \leftarrow 2n$ (*B Stores branch number of* M)
$r \leftarrow \left\lfloor \frac{n+1}{2} \right\rfloor$
for $(k \leftarrow 1 \text{ to } r)$ **do**
  while $(T_l \neq \varnothing)$ **do**
    Choose $x \in T_l$
    $weight \leftarrow w_h(x)+w_h(M.x)$
    if $(weight < B)$ **then**
      $B \leftarrow weight$
    end if
    $weight \leftarrow w_h(x)+w_h(M^{-1}.x)$
    if $(weight < B)$ **then**
      $B \leftarrow weight$
    end if
    $T_l \leftarrow T_l / \{x\}$
  end while
end for
return B
end function

Next, we study the CC of Algorithm 1. The computationally dominating steps in the algorithm are $(M.x)$ and $(M^{-1}.x)$ (i.e., multiplication of matrix M by x and M^{-1} by x). This step is getting repeated $\sum_{l=1}^{r}|T_l|$ times, where $r=\left\lfloor \frac{n+1}{2} \right\rfloor$ and $|T_l| = \binom{n}{l}\mu^l$, where $\mu+1=2^m$.

As T_l contains vectors of weight l from $(GF(2^m))^n$. Given $1\leq l\leq r$, the steps $(M.x)$ and $(M^{-1}.x)$ involve matrix multiplication of order $n\times n$ with an element of x in T_l. As a result, the CC of Algorithm 1 is:

$$2n\sum_{l=1}^{r}|T_l|.l = 2n\sum_{l=1}^{\left\lfloor \frac{n+1}{2} \right\rfloor}\binom{n}{l}\mu^l. \tag{1}$$

Next, we observe the CC of Algorithm 1 for involutory matrices. As $M=M^{-1}$ for an involutory matrix M of order $n\times n$, its branch number d can be determined by searching for the minimum value of $w_h+w_h(M.x)$ among the input vectors of weight up to $d/2$, $d \le n+1$. Hence, the CC of Algorithm 1 for involutory matrices is:

$$n\sum_{l=1}^{r}|T_l|.l = n\sum_{l=1}^{\left\lfloor\frac{n+1}{2}\right\rfloor}\binom{n}{l}l\mu^l, \quad \text{where} \quad \mu+1=2^m. \tag{2}$$

Using Eqn. (2), the CC for computing the branch number of the 8×8 involutory matrix M over $GF(2^8)$ is approximately 2^{44}. In the next subsection, we show that in the case of specific matrices such as Hadamard matrices, the CC of Algorithm 1 is by a factor of n.

3.2.2 Reduction of Computational complexity for Hadamard Matrices

For an $n\times n$ Hadamard matrix, our approach reduces the complexity of Algorithm 1 by a factor of $1/n$. We present our main result as follows.

Theorem 2. Assume H is an $n\times n$ Hadamard matrix over $GF(2^m)$. Let ρ^T be a column vector in $(GF(2^m))^n$. Then:

$$wt(H.\rho^T) = wt(H.\phi_z(\rho)^T),$$

where $\phi_z : (GF(2^m))^n \to (GF(2^m))^n$ for each $z=1,2,\ldots,n$ is a map that permutes the elements of the vector ρ and is defined as:

$$\phi_z(\rho) = (a_z, a_{z\oplus 1}, \ldots, a_{z\oplus(n-1)}),$$
$$\rho = (a_0, a_1, \ldots, a_{n-1}) \in (GF(2^m))^n.$$

We discuss an important result in Lemma 1 to prove Theorem 1. This lemma will be used to prove Theorem 1.

Lemma 1. Let $\rho = (a_0, a_1, \ldots, a_{n-1})$, $\eta = (a_0, a_1, \ldots, a_{n-1}) \in (GF(2^m))^n, \varsigma = \{1,2,\ldots,n-1\}$ and $\Theta : (GF(2^m))^n \times (GF(2^m))^n \to (GF(2^m))^n$ is a map defined as:

$$\Theta(\rho,\eta) = \left(\sum_{i\in\varsigma} a_i b_i, \sum_{i\in\varsigma} a_{i\oplus 1} b_i, \ldots, \sum_{i\in\varsigma} a_{i\oplus(n-1)} b_i\right).$$

Then, we have:

$$wt(\Theta(\rho,\eta)) = wt(\Theta(\rho,\phi_z(\eta))), \qquad 0 \le z \le n, \tag{3}$$

where, ϕ_z is the same as defined in Theorem 1.

Proof. We observe that $\Theta(\phi_j(\rho),\eta)$ is a permutation of entries of $\Theta(\rho,\eta)$ for every $0 \le j < n$. Therefore, we can write

$$wt(\Theta(\rho,\eta)) = wt(\Theta(\phi_z(\rho),\eta)), \qquad 0 \le j < n.$$

Further, we observe that

$$\Theta(\rho,\phi_z(\eta)) = \left(\sum_{i\in\varsigma} a_i b_{i\oplus z}, \sum_{i\in\varsigma} a_{i\oplus 1} b_{i\oplus z}, \ldots, \sum_{i\in\varsigma} a_{i\oplus(n-1)} b_{i\oplus z}\right).$$

For $0 \le l < n$, the above expression may be rewritten as:

$$\Theta(\rho,\phi_l(\eta)) = \left(\sum_{i\in\varsigma} a_i b_{i\oplus l}, \sum_{i\in\varsigma} a_{i\oplus 1} b_{i\oplus l}, \ldots, \sum_{i\in\varsigma} a_{i\oplus(n-1)} b_{i\oplus l}\right).$$

Let $z\oplus l=m$ for some $z, m \in \varsigma$. Then, we derive that:

$$\Theta(\rho,\phi_l(\eta)) = \left(\sum_{i\in\varsigma} a_i b_{i\oplus l}, \sum_{i\in\varsigma} a_{i\oplus 1} b_{i\oplus l}, \ldots, \sum_{i\in\varsigma} a_{i\oplus(n-1)} b_{i\oplus l}\right).$$

Furthermore, let $m\oplus i=h$. Consequently, we reach at:

$$\Theta(\rho,\phi_l(\eta)) = \left(\sum_{m\oplus h\in\varsigma} a_{m\oplus h} b_{z\oplus h}, \sum_{m\oplus h\in\varsigma} a_{m\oplus h\oplus 1} b_{z\oplus h}, \ldots, \sum_{m\oplus h\in\varsigma} a_{m\oplus h\oplus(n-1)} b_{z\oplus h}\right).$$

Since m is fixed, $h \oplus m \in \varsigma$ is the same as $h \in \varsigma$. Therefore, we further obtain that:

$$\Theta(\rho,\phi_l(\eta)) = \left(\sum_{h\in\varsigma} a_{m\oplus h} b_{z\oplus h}, \sum_{h\in\varsigma} a_{m\oplus h\oplus 1} b_{z\oplus h}, \ldots, \sum_{h\in\varsigma} a_{m\oplus h\oplus(n-1)} b_{z\oplus h}\right) = \Theta(\phi_m(\rho),\phi_z(\eta)). \tag{4}$$

By combining Eqn. (3) and Eqn. (4) we conclude that

$$wt(\Theta(\rho,\phi_l(\eta)) = wt(\Theta(\phi_m(\rho),\phi_z(\eta)) = wt(\Theta(\rho,\phi_z(\eta))). \tag{5}$$

Thus, the result holds. To this end, we prove Theorem 2.

Proof of Theorem 2. Consider the Hadamard matrix H formed by permutations of elements of ρ as:

$$H = (\phi_0(\rho), \phi_1(\rho), \ldots, \phi_{n-1}(\rho))^T.$$

For an arbitrary vector $\eta \in (GF(2^m))^n$, it holds that:

$$H.\eta^T = \Theta(\rho,\eta)^T.$$

Using Eqn. (5) of Lemma 1, we note that:

$$wt(H.\eta^T) = wt(\Theta(\rho,\eta)^T) =$$
$$wt(\Theta(\rho,\phi_z(\eta))^T) = wt(H.\phi_z(\eta)^T).$$

This completes the proof.

It follows from Theorem 1 that post-multiplying a Hadamard matrix H with a permutation of a column vector ρ^T determined by ϕ_z, $0 \le z < n$, will not change the weight of the product of H and ρ^T. It indicates that the search space of input vectors for computing the branch number of an $n\times n$ Hadamard matrix H can be reduced by a factor of $1/n$. More precisely, it becomes:

$$n\sum_{l=1}^{r}|T_l|.l.\frac{1}{n} = \sum_{l=1}^{\left\lfloor\frac{n+1}{2}\right\rfloor}\binom{n}{l}l\mu^l, \quad \text{where} \quad \mu+1=2^m.$$

Finally, we give an algorithm for computing r_2 and derive its CC, where r_2 represents the number of 8×8 IMDS matrix representatives. We recall that r_2 equals the number of Hadamard matrix forms M_2 with a branch number of 9. The total matrices of the form M_2 are $(2^m-1)^3$. Let M_2' be a set containing all matrices of the form M_2. Clearly, $M_2' = (2^m-1)^3$.

Algorithm 2
Computation of r_2 in $GF(2^m)$

$t \leftarrow (2^m - 1)^3$
$j \leftarrow 0$
for $(k \leftarrow 1 \text{ to } t)$ **do**
  Choose $M \in M_2'$
  $b \leftarrow branch\ number(M)$ (*Compute branch number b of M using Algorithm 1.*)

If (b=9) then
$j \leftarrow j+1$
end if
end for
return j

Therefore, the CC of Algorithm 2 is:

$$\sum_{l=1}^{\left\lfloor \frac{n+1}{2} \right\rfloor} \binom{n}{l} l\mu^{l+3}, \quad \text{where} \quad \mu + 1 = 2^m.$$

3.3 Generation of 4×4 and 8×8 Non-IMDS Hadamard Matrices

The aim of this subsection is different from the previous subsections. In earlier subsections, we have generated $2^k \times 2^k$ IMDS matrices for LC for $k = 2, 3$. In this subsection, we generated $2^k \times 2^k$ lightweight non-IMDS Hadamard matrices for $k = 2, 3$. We generated these matrices by exhaustively computing the branch number using Algorithm 1. First, we choose elements of the first row with a lower XOR count. Then by computing its branch number, we generate lightweight non-IMDS Hadamard matrices, these generated matrices are significantly lighter than the non-IMDS matrix used in WHIRLPOOL[25]. The experimental results and the generated matrices are given in the next section.

4. EXPERIMENTAL RESULTS

Using our proposed method, we create lightweight IMDS and non-IMDS matrices of order $2^k \times 2^k$ over $GF(2^4)$ and $GF(2^8)$, respectively for $k = 2, 3$ (Tables 2-4). In tables, MT denotes matrix type, FF denotes finite field, R_1 denotes the first-row elements, XC denotes the XOR count and Ref denotes the reference(s).

Table 2. 4×4 MDS Matrices Over *GF*(2^4)

MT	FF	R_1	XC	Ref
Involutory Hadamard	$GF(2)[x]/(0x13)$	$(1_x, 4_x, 9_x, d_x)$	$6+3\times4=18$	12, 30
Involutory Hadamard	$GF(2)[x]/(0x19)$	$(1_x, 2_x, 6_x, 4_x)$	$6+3\times4=18$	12, 31
Hadamard	$GF(2)[x]/(0x13)$	$(1_x, 2_x, 4_x, 9_x)$	$4+3\times4=16$	Our result
Hadamard	$GF(2)[x]/(0x13)$	$(1_x, 2_x, 8_x, 9_x)$	$5+3\times4=17$	12

8×8 MDS Matrices Over *GF*(2^4)

MT	FF	R_1	XC	Ref
Involutory Hadamard	$GF(2)[x]/(0x13)$	$(f_x, a_x, 8_x, 5_x, 3_x, c_x, 2_x, 4_x)$	$36+7\times4=64$	Our result
Hadamard	$GF(2)[x]/(0x13)$	$(8_x, d_x, 9_x, 2_x, c_x, 1_x, 6_x, a_x)$	$26+7\times4=54$	Our result
Hadamard	$GF(2)[x]/(0x13)$	$(5_x, 4_x, a_x, 6_x, 2_x, d_x, 8_x, 3_x)$	$33+7\times4=61$	29
Hadamard	$GF(2)[x]/(0x13)$	$(5_x, e_x, 4_x, 7_x, 1_x, 3_x, f_x, 8_x)$	$39+7\times4=67$	29

Table 3. 4×4 IMDS Matrices Over *GF*(2^8)

MT	FF	R_1	XC	Ref
Hadamard	$GF(2)[x]/(0x1c3)$	$(01_x, 02_x, c1_x, c3_x)$	$18+3\times8=42$	Our result
Hadamard	$GF(2)[x]/(0x1c3)$	$(01_x, 91_x, 70_x, e1_x)$	$18+3\times8=42$	Our result
Hadamard	$GF(2)[x]/(0x165)$	$(01_x, 02_x, b0_x, b2_x)$	$16+3\times8=40$	12
Hadamard	$GF(2)[x]/(0x11d)$	$(01_x, 02_x, 04_x, 06_x)$	$22+3\times8=46$	26
Hadamard	$GF(2)[x]/(0x11d)$	$(01_x, 08_x, 02_x, 0a_x)$	$30+3\times8=54$	27
Compact Cauchy	$GF(2)[x]/(0x11b)$	$(01_x, 12_x, 04_x, 16_x)$	$54+3\times8=78$	6
Hadamard Cauchy	$GF(2)[x]/(0x11b)$	$(01_x, 02_x, fc_x, fe_x)$	$74+3\times8=98$	32

8×8 IMDS Matrices Over *GF*(2^8)

MT	FF	R_1	XC	Ref
Hadamard	$GF(2)[x]/(0x1c3)$	$(01_x, 02_x, 03_x, 70_x, 04_x, 91_x, 05_x, e1_x)$	$46+7\times8=102$	Our result
Hadamard	$GF(2)[x]/(0x1c3)$	$(01_x, 02_x, 08_x, 38_x, 48_x, 91_x, e1_x, 0a_x)$	$52+7\times8=108$	Our result
Hadamard	$GF(2)[x]/(0x1c3)$	$(01_x, 02_x, 03_x, 91_x, 04_x, 70_x, 05_x, e1_x)$	$46+7\times8=102$	12
Hadamard	$GF(2)[x]/(0x11d)$	$(01_x, 03_x, 04_x, 05_x, 06_x, 08_x, 0b_x, 07_x)$	$100+7\times8=156$	28
Hadamard Cauchy	$GF(2)[x]/(0x11d)$	$(01_x, 02_x, 06_x, 8c_x, 30_x, fb_x, 87_x, c4_x)$	$122+7\times8=178$	32

Table 4. 4×4 Non-IMDS Matrices Over $GF(2^8)$

MT	FF	R_1	XC of first row	Ref
Hadamard	$GF(2)[x]\,/\,(0x1c3)$	$(e1_x, 01_x, 04_x, 91_x)$	$13+3\times8=37$	Our result
Hadamard	$GF(2)[x]\,/\,(0x1c3)$	$(01_x, 02_x, 04_x, 91_x)$	$13+3\times8=37$	12
Circulant	$GF(2)[x]\,/\,(0x11b)$	$(02_x, 03_x, 01_x, 01_x)$	$14+3\times8=38$	24

8×8 Non-IMDS Matrices Over $GF(2^8)$

MT	FF	R_1	XC of first row	Ref
Hadamard	$GF(2)[x]\,/\,(0x1c3)$	$(01_x, 02_x, e0_x, 08_x, e1_x, a9_x, 04_x, 91_x)$	$40+7\times8=96$	Our result
Hadamard	$GF(2)[x]\,/\,(0x1c3)$	$(01_x, 02_x, 03_x, 08_x, 04_x, 91_x, e1_x, a9_x)$	$40+7\times8=96$	12
Circulant	$GF(2)[x]\,/\,(0x11d)$	$(01_x, 01_x, 04_x, 01_x, 08_x, 05_x, 02_x, 09_x)$	$49+7\times8=105$	25

Table 5. Distribution of XOR(*a*) values with respect to $a \in GF(2^8)$ corresponding to two generator polynomials $0x1c3$ and $0x11d$.

x	$0x1c3$	$0x11d$	x	$0x1c3$	$0x11d$	x	$0x1c3$	$0x11d$	x	$0x1c3$	$0x11d$	x	$0x1c3$	$0x11d$
0x00	0	0	..4	27	24	..8	29	26	..c	31	22	0xd0	31	28
..1	0	0	..5	33	30	..9	35	32	..d	27	22	..1	29	32
..2	3	3	..6	26	19	..a	36	29	..e	32	15	..2	36	31
..3	9	11	..7	34	25	..b	40	35	0x9f	26	15	..3	32	35
..4	5	6	..8	11	23	..c	26	16	0xa0	19	29	..4	38	28
..5	11	14	..9	19	29	..d	34	22	..1	17	27	..5	34	32
..6	10	13	..a	22	22	..e	35	23	..2	26	30	..6	41	35
..7	14	21	..b	28	28	0x6f	41	29	..3	22	28	..7	35	39
..8	7	09	..c	24	33	0x70	10	23	..4	28	21	..8	26	13
..9	11	17	..d	30	39	..1	18	21	..5	24	19	..9	2	17
..a	12	16	..e	29	32	..2	19	32	..6	29	26	..a	33	20
..b	18	24	0x3f	33	38	..3	25	30	..7	23	24	..b	29	24
..c	14	15	0x40	20	19	..4	21	21	..8	14	20	..c	35	21
..d	20	23	..1	24	15	..5	27	19	..9	08	18	..d	31	25
..e	13	22	..2	23	16	..6	28	26	..a	23	25	..e	40	28
0x0f	21	30	..3	29	12	..7	32	24	..b	19	23	0xdf	38	32
0x10	12	12	..4	25	17	..8	25	34	..c	25	12	0xe0	09	24
..1	18	12	..5	31	13	..9	29	32	..d	21	10	..1	03	30
..2	11	21	..6	26	10	..a	28	39	..e	28	17	..2	16	19
..3	19	21	..7	34	06	..b	34	37	0xaf	26	15	..3	12	25
..4	13	18	..8	11	30	..c	30	32	0xb0	21	29	..4	18	32
..5	17	18	..9	19	26	..d	36	30	..1	17	35	..5	14	38
..6	18	27	..a	20	27	..e	31	37	..2	24	32	..6	23	31
..7	24	27	..b	26	23	0x7f	39	35	..3	22	38	..7	21	37
..8	17	17	..c	22	24	0x80	25	21	..4	18	21	..8	20	21
..9	23	17	..d	28	20	..1	21	29	..5	12	27	..9	18	27
..a	22	22	..e	29	21	..2	26	16	..6	27	24	..a	25	16
..b	26	22	0x4f	33	17	..3	20	24	..7	23	30	..b	21	22
..c	12	23	0x50	18	25	..4	24	17	..8	22	24	..c	27	25
..d	20	23	..1	24	29	..5	22	25	..9	18	30	..d	23	31
..e	23	32	..2	25	20	..6	31	12	..a	23	31	..e	30	20
0x1f	29	32	..3	29	24	..7	27	20	..b	17	37	0xef	24	26
0x20	16	16	..4	15	19	..8	30	20	..c	21	24	0xf0	25	34
..1	22	14	..5	23	23	..9	26	28	..d	19	30	..1	21	32
..2	21	13	..6	24	14	..a	33	15	..e	28	27	..2	28	31
..3	25	11	..7	30	18	..b	31	23	0xbf	24	33	..3	22	29
..4	11	26	..8	19	28	..c	27	12	0xc0	27	20	..4	26	38
..5	19	24	..9	25	32	..d	21	20	..1	23	16	..5	24	36
..6	22	23	..a	20	23	..e	36	03	..2	32	25	..6	31	35
..7	28	21	..b	28	27	0x8f	32	11	..3	30	21	..7	27	33
..8	17	21	..c	22	26	0x90	11	35	..4	26	28	..8	30	31
..9	23	19	..d	26	30	..1	05	35	..5	20	24	..9	26	29
..a	16	18	..e	25	25	..2	20	28	..6	33	33	..a	35	28
..b	24	16	0x5f	31	29	..3	16	28	..7	29	29	..b	33	26
..c	18	27	0x60	24	19	..4	22	27	..8	28	17	..c	29	39
..d	22	25	..1	30	25	..5	18	27	..9	24	13	..d	23	37
..e	23	28	..2	25	26	..6	25	24	..a	31	26	..e	36	32
0x2f	29	26	..3	33	32	..7	23	24	..b	25	22	0xff	32	30
0x30	20	18	..4	27	17	..8	22	26	..c	29	25			
..1	24	24	..5	31	23	..9	20	26	..d	27	21			
..2	25	17	..6	30	24	..a	29	19	..e	34	30			
..3	31	23	..7	36	30	..b	25	19	0xcf	30	26			

Notation: (here, ...1 written below 0x00 means 0x01 and so on. Similarly, ...1 written below 0x00 means 0x01. This notation is used throughout in the table)

We create lightweight $2^k \times 2^k$ IMDS matrices over $GF(2)[x]/(0x13)$ and $GF(2)[x]/(0x1c3)$ for $k = 2,3$ in terms of XOR count. We show that our IMDS matrices are significantly lighter than the matrices used in ANUBIS[26], CLEFIA[27], and KHAZAD[28]. We also create some non-IMDS matrices that are lighter than the non-IMDS matrix utilized in WHIRPOOL[25].

Our paper's findings are as follows:

- We created a lightweight 4×4 non-IMDS Hadamard matrix in the finite field $GF(2)[x]/(0x13)$ with XOR count 16, which is the minimum XOR count among the other best-known XOR counts of lightweight non-IMDS matrices over $GF(2^4)$ (Table 2). In addition, we generated 4×4 IMDS Hadamard and non-IMDS Hadamard matrices with optimal XOR counts. Table 2 also demonstrates that the XOR count is affected by the polynomial used to generate a finite field.
- We created an 8×8 IMDS Hadamard matrix in $GF(2)[x]/(0x13)$ with an XOR count of 64 (Table 2). To the best of our knowledge, this is the best XOR count for constructing an 8×8 MDS matrix in the field $GF(2)[x]/(0x13)$. In addition, we created an 8×8 non-IMDS Hadamard matrix in $GF(2)[x]/(0x13)$ with XOR count 54 (Table 2). This XOR count is the minimum of the known XOR counts of 8×8 non-IMDS Hadamard matrices generated[29].
- We created a lightweight 4×4 IMDS Hadamard matrix in $GF(2)[x]/(0x1c3)$ with an XOR count of 40 (Table 3). This XOR count is less than that of the 4×4 IMDS Hadamard matrices used in the well-known ANUBIS and CLEFIA ciphers. In addition, we created a lightweight 8×8 IMDS Hadamard matrix in $GF(2)[x]/(0x1c3)$ with XOR count 102 (Table 2). This XOR count is equal to the minimum known XOR count required in the implementation of an 8×8 IMDS Hadamard matrix in $GF(2)[x]/(0x1c3)$ (Table 3 and [12]).
- In addition, we created lightweight 4×4 and 8×8 non-IMDS Hadamard matrices over $GF(2^8)$ having XOR counts of 37 and 96, respectively (Table 4). These XOR counts are equal to the minimum known XOR counts of lightweight 4×4 and 8×8 non-IMDS Hadamard matrices (see [12]). We constructed a 4×4 non-IMDS Hadamard matrix that requires fewer XOR operations than the well-known circulant matrix associated with AES (see [24]).
- The MDS matrices generated in[16] can also be generated using the matrix forms M_1 and M_2 proposed in section 3. For example, if we take $\beta_0 = 0x2$ and $\beta_1 = 0x5$ in M_1 then we can obtain Hadamard matrix $had(0x1, 0x2, 0x5, 0x7)$, which can then be utilized to build the same GHadamard matrix considered in Example 5 of[16]. Similarly, using our matrix forms M_1 and M_2, we can construct the other GHadamard matrices mentioned in Examples 6 and 7 of[16].
- The XOR count of the 4×4 IMDS Hadamard matrix over $GF(2)[x]/(0x11d)$, which is used in ANUBIS, is 46. Furthermore, the XOR count of CLEFIA's 4×4 IMDS Hadamard matrix over $GF(2)[x]/(0x11d)$, is 54. Table 2 shows that the 4×4 IMDS Hadamard matrix over $GF(2)[x]/(0x1c3)$ generated in this study is significantly lighter than the matrices used in the ANUBIS and CLEFIA ciphers. Furthermore, the XOR count of the 8×8 IMDS Hadamard matrix over $GF(2)[x]/(0x11d)$, used in the KHAZAD cipher is 156. In this paper, we created an 8×8 IMDS Hadamard matrix over $GF(2)[x]/(0x1c3)$ with a much lower XOR count than the matrix used in KHAZAD.

5. CONCLUSIONS

Author presented two new Hadamard matrix forms for generating $2^k \times 2^k$ IMDS matrices over $GF(2^m)$, for LC, where k=2,3 respectively. The proposed methods provide a straightforward way for generating 4×4 IMDS matrices as well as a hybrid method for constructing 8×8 IMDS matrices over $GF(2^m)$. In addition, we provided an algorithm for computing the branch number of $n \times n$ invertible matrices over $GF(2^m)$. In the case of Hadamard matrices, we reduced the computational complexity of our algorithm. Furthermore, we generated 4×4 and 8×8 non-IMDS matrices for LC with optimal XOR counts using the branch number algorithm. In future, proposed techniques will be useful for search-based methods developed in prior studies to generate IMDS matrices over $GF(2^m)$.

REFERENCES

1. Shannon, C.E. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 1949, **28**, 656-715. doi: 10.1002/j.1538-7305.1949.tb00928.x
2. Guo, Z.; Wu, W. & Gao, S. Constructing lightweight optimal diffusion primitives with Feistel structure. Selecting Areas in Cryptography, 2015, 352-372. doi: 10.1007/978-3-319-31301-6_21
3. Biham, E. & Shamir, A. Differential cryptanalysis of DES-like cryptosystems, *J. Cryptol.*, 1990, **4**, 3-72. doi: 10.1007/BF00630563
4. Matsui, M. Linear cryptanalysis method for DES cipher, EUROCRYPT 93, Advances in Cryptology, 1994, 386-397. doi: 10.1007/3-540-48285-7_33
5. Yang, Y.; Zeng, X. & Wang, S. Construction of lightweight involutory MDS matrices, *Des. Codes Cryptog.,* 2021, **89**, 1453-1483. doi: 10.1007 /s1 0623-021-00879-3
6. Cui, T.; Jin, C. & Kong, Z. On compact cauchy matrices for substitution permutation networks, *IEEE Trans. Computers.*, 2015, **64**(7), 2098-2102. doi: 10. 1109/TC.2014.2346180
7. Mousavi, M.; Zaghian, A. & Esmaeili, M. Involutory-multiple-lightweight MDS matrices based on Cauchy-type matrices. *Adv. Math. Commun*., 2021, **15**(4), 589-610. doi:10.3934/amc.2020084
8. Sajadieh, M.; Dakhilalian, M.; Mala, H. & Omoom, B. On construction of involutory mds matrices from vandermonde matrices in GF(2^q). *Des. Codes Cryptogr.*, 2012, **64**(3), 287-308. doi: 10.1007/s10623-011-9578-x
9. Gupta, K.C. & Ray, I.G. On constructions of MDS matrices from companion matrices for lightweight cryptography, CD-ARES 2013, LNCS, 8128, 29-43. doi: 10.1007/978-3-642-40588-4_3

10. Sajadieh, M. & Mohsen, Mousavi M. Construction of MDS matrices from generalised Feistel structures. *Des. Codes Cryptogr*., 2021, **89**(7), 1433-1452. doi: 10.1007/s10623-021-00876-6
11. Gupta, K.C.; Pandey, S. K.; Ray, I.G. & Samanta, S. Cryptographically significant Mds matrices over finite fields: A brief survey and some generalized results. *Adv. Math. Commun*., 2019, **13**(4), 779-843. doi: 10. 3934/amc.2019045
12. Sim, S.M.; Khoo, K.; Oggier, F. & Peyrin, T. Lightweight MDS involution matrices. In: Leander, G.(eds) Fast Software Encryption, 2015, 471-493. doi: 10.10 07/978-3-662-48116-5_23
13. Sajadieh, M.; Dakhilalian, M.; Mala, H. & Sepehrdad, P. Recursive diffusion layers for block ciphers and hash functions. In: Canteaut, A.(eds) fast software encryption, 2012, 385-401. doi: 10.1007/978-3-642-34047-5_22
14. Wu; S.; Wang, M. & Wu, W. Recursive diffusion layers for (lightweight) block ciphers and hash functions. *In*: Knudsen, L.R., Wu, H. (eds) Selected areas in cryptography, 2012, 355-371. doi: 10.1007/978-3-642-35999-6_23
15. Xiang, Z.; Zeng, X.; Lin, D.; Bao, Z. & Zhang, S. Optimising implementations of linear layers. *IACR Trans. Symmetric Cryptol.*, 2020, 120-145. doi:10.13154/tosc.v0.i0.0-0
16. Pehlivanoglu, M.K.; Sakall, M.T.; Akleylek, S.; Duru, N. & Rijmen, V. Generalisation of hadamard matrix to generate involutory MDS matrices for lightweight cryptography. *IET Inf. Secur.*, 2018, **12**(4), 348-355. doi: 10.1049/iet-ifs.2017.0156
17. Yongqiang, Li. & Mingsheng, W. On the construction of lightweight circulant involutory MDS matrices. *IACR Trans. Symmetric Cryptol.,* 2016, **1**, 121-139. doi: 10.1007/978-3-662-52993-5_7
18. Mohsenifar, N. & Sajadieh, M. Introducing a new connection between the entries of MDS matrices constructed by generalized Cauchy matrices in GF(2^q). *J. Appl. Math. Comput.,* 2023. doi: 10.1007/s12190-023-01907-2
19. Sarkar, S. & Sim, S. A deeper understanding of the XOR count distribution in the context of lightweight cryptography, AFRICACRYPT 16, 2016, LNCS, **9646**, 167-182. doi: 10.1007/978-3-319-31517-1_9
20. Guzel, G.G.; Sakalli, M.T.; Akleylek, S.; Rijmen, V. & Engellenmis, Y. C. A new matrix form to generate All 3×3 Involutory MDS matrices over F_2^m. *Inf. Process. Lett.,* 2019, **147**, 61-68. doi:10.1016/j.ipl.2019.02.013
21. Altawy, R. & Youssef, A. Preimage analysis of the Maelstrom-0 hash function. *In* Security, Privacy, and Applied Cryptography Engineering, 2015,113-126. doi: 10.1007/978-3-319-24126-5_7
22. Dai, W.; Soichi, F.; Hirotaka, Y.; Kazuo, T.; & Bart, P. A new keystream generator MUGI, fast software encryption. 2002, 179-184. doi: 10.1007/3-540-45661-9_14
23. Lidl, R. & Niederreiter, H. Finite fields. Cambridge University Press, Cambridge, 1983. doi: 10.1007/978-3-662-04722-4
24. Daemen, J. & Rijmen, V. The design of rijndael. AES-The Advanced Encryption Standard, 2002. doi: 10.1007/978-3-662-04722-4
25. Barreto, P.S.L.M. & Rijmen, V. The Whirlpool hashing function. *In* Proceedings of the first NESSIE Workshop, 2000, 15. https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions.html (Accessed on 16 January 2023).
26. Barreto, P.S.L.M. & Rijmen, V. The Anubis block cipher, Submission to the NESSIE Project, 2000. https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions.html (Accessed on 30 January 2023).
27. Shirai, T.; Shibutani, K.; Akshita, T.; Moriai, S.; & Iwata, T. The 128-Bit block cipher CLEFIA (Extended Abstract) International workshop fast software encryption. Lecture Notes in computer science, 2007, 181-195. doi: 10.1007/978-3-540-74619-5_12
28. Barreto, P.S.L.M. & Rijmen, V. The khazad legacy-level block cipher. NESSIE Algorithm Submission, 2000. https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions.html (Accessed on 11 February 2023).
29. Barreto, P.S.L.M.; Ventzislav, Nikova.; Svetla, Nikova.; Vincent, Rijmen. & Tischhauser, E. Whirlwind: A new cryptographic hash function. *Des. Codes Cryptogr.*, 2010, **56**(2-3), 141-162. doi: 10.1007/s10623-010-9391-y
30. Jean, J.; Nikolic, I. & Peyrin, T. Joltik v1.1. Submission to the CAESAR competition, 2014. https://competitions.cr.yp.to/round1/joltikv1.pdf (Accessed on 6 March 2023).
31. Kavun, E.B.; Lauridsen, M.; Leander, G.; Rechberger, C.; Schwabe, P. & Yalcin, T. Prost v1.1. Submission to the CAESAR Competition, 2014. https://competitions. cr.yp.to/round1/proestv1.pdf (Accessed on 20 March 2023).
32. Gupta, K.C. & Ray, I.G. On constructions of involutory MDS matrices. *In* AFRICACRYPT 2013, pp. 43-60. doi: 10.1007/978-3-642-38553-7_3

CONTRIBUTORS

Mr Yogesh Kumar obtained his MSc from CCS University, Meerut. He has been working as a Scientist in DRDO-SAG, Delhi, India. His area of interest include: Algebra and cryptography.
He has contributed in conceptualisation and compared the result obtained and findings with available in the literature as of today. Also, he has prepared the first draft of the paper.

Dr P.R. Mishra obtained his PhD from Banaras Hindu University. He has been working as a Scientist in DRDO-SAG, Delhi, India. His area of interest include: Algebra, number theory and cryptography.
He has contributed in this paper by discussing all the key aspects. His contribution to this paper as a scientific point of view and shaping the results and findings as presented.

Dr Atul Gaur obtained his PhD from IIT Kanpur. He has been working as an Associate Professor in Mathematics Department, University of Delhi, India. His area of interest include: Commutative algebra, multiplication modules and graph theory.

In the current study he has guided in editorial quality, reviewed the progress of work and provided valuable suggestions.

Dr Gaurav Mittal obtained his PhD from IIT Roorkee. He has been working as a Scientist at DRDO-JCB, Delhi, India. His area of interest include: Algebra and cryptography.

In this study, he has given the idea, reviewed the results, continuously provided the guidance and given many valuable inputs. He carried out existing literature survey related to this study and also done sequencing, drafting and editing.