IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2025, No. 3, pp. 868–890. DOI:10.46586/tosc.v2025.i3.868-890

New General MDS Matrix Construction Method Towards Low Area

Yan He[1], Tingting Cui[1,2(✉)], Qing Ling[1] and Xi Han[1]

[1] School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China
[2] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao, China
yan_he@hdu.edu.cn,cuitingting@hdu.edu.cn,lq1003hzhf@126.com,hanxi@hdu.edu.cn

Abstract. Maximum Distance Separable (MDS) matrices have been widely used in symmetric cryptographic primitives because of their excellent cryptographic properties. However, due to the heavy area cost, larger-scale MDS matrices than 4×4 ones are limited in ciphers, although they have a larger branch number. In this paper, we propose a general method for constructing MDS matrices with low implementation area cost, using matrix decomposition, automatic search, and symbolic computation techniques. According to matrix decomposition theory, every invertible matrix can be decomposed into a sequence of elementary matrices, including Type-1 (row switching), Type-2 (row multiplication) and Type-3 (row addition) elementary matrices. So, we first propose a greedy algorithm to construct MDS matrix patterns with as few Type-3 elementary matrices as possible. Then, we build an automatic search model to minimize the implementation area of multiplication coefficients used in Type-3 elementary matrices. Lastly, another greedy strategy is raised to further reduce the implementation area of MDS matrix patterns by introducing a few Type-2 elementary matrices. In comparison to previous methods, our approach is more general and effective for constructing lower-area MDS matrices. To demonstrate the efficiency of our method, we apply the framework on constructing $m \times m$ MDS matrices over $\mathbb{F}_{2^n}$ or $GL(n, \mathbb{F}_2)$, where $m \in \{4, 5, 6, 7, 8\}$ and $n \in \{4, 8, 16, 32, 64\}$. The 4×4 MDS matrices constructed by our method can also reach the minimum area. The 5×5, 6×6 and 7×7 MDS matrices constructed by our method have lower area compared to previous ones. While the 8×8 MDS matrices with $n \in \{16, 32, 64\}$ constructed by our method also have lower area compared to previous ones.

Keywords: MDS matrix · Matrix decomposition · Symbolic computation · Automatic optimization · Low area

1 Introduction

1.1 Background

Linear diffusion layers are very important linear components in symmetric cryptographic primitives, including block ciphers, hash functions, authenticated encryption schemes and message authentication codes. As one type of linear diffusion layers, Maximum Distance Separable (MDS) matrices have been widely used in international standard ciphers such as AES [DR02], SM4 [ISO21], CLEFIA [KM11] and so on. However, due to the heavy area cost, large-scale MDS matrices are limited in ciphers, although they have large branch numbers. Therefore, designing MDS matrices with low area has become a hot research topic in recent years, especially for post-quantum secure cryptographic primitives.

Previous effective methods available to construct MDS matrices can be mainly classified into three classes:

Received: 2025-06-01 Accepted: 2025-08-01 Published: 2025-09-25

- determine an MDS matrix directly by a special mathematical matrix;
- construct an MDS matrix by iterating a special structure several times;
- exhaustively search an MDS matrix by combining some basic operations.

The first class of methods generally takes current mathematical matrices such as Circulant, Hadamard, Toeplitz and Cauchy matrices [SDMO12, GR13, LW16, SS16, SS17, GPV17, PSA$^+$18, CL19] directly as MDS matrices. The main advantage of this class of methods is that it allows easy construction of large-scale MDS matrices due to the compact search space. However, their main disadvantage is that the implementation area of these mathematical matrices is noticeably larger than the ones produced by other classes of methods. The second class of methods employs iterative constructions, in which certain special structures, such as LFSR [SDMS12, WWW13], GFS [WWW13], DSI [TTKS18, LSS$^+$19] and DLS [GPS22], are repeatedly applied to construct MDS matrices. This class of methods is helpful for constructing low-area MDS matrices, but it is difficult to reach the minimum area bound, as it heavily depends on iterative structures. To break the iterative feature, Sajadieh and Mousavi [SM21] proposed the EGFS construction, which produced 8×8 MDS matrices with the current minimum area. A further limitation is that some constructions, such as GFS and EGFS, can only generate even-dimensional MDS matrices. The third class of methods, i.e., exhaustively searching MDS matrices by combining some basic operations, is one type of interesting and more general method. In [DL18], Duval and Leurent proposed a construction method based on three operations, including XOR, COPY and Multiplication over a ring. It obtained the minimum area for 4×4 MDS matrices. In [WLTZ23], Wang et al. proposed another construction method based on three types of elementary matrices over a field, inspired by the optimization methods for existing matrices based on matrix decomposition technique [XZL$^+$20, LXZZ21, YWS$^+$24]. It also achieved the minimum area of 4×4 MDS matrices. However, both above methods can only construct 4×4 MDS matrices due to the huge search space.

Matrix decomposition is an attractive technique. Every invertible matrix can be decomposed into a sequence of elementary matrices. There are three types of elementary matrices, including Type-1 (switch two rows), Type-2 (multiply a row by a non-zero number) and Type-3 (add a multiple of one row to another row) elementary matrices. Compared to the three classes of traditional methods, it is a good idea to use the matrix decomposition technique to construct larger-scale MDS matrices than 4×4 ones, but we need to solve the following problems:

- It is infeasible to exhaustively search all combinations of three types of elementary matrices to construct $m \times m$ MDS matrices as the dimension m increases. Therefore, how to minimize the number of Type-3 elementary matrices so as to reduce the implementation area while maintaining the MDS property?
- Each Type-2 or Type-3 elementary matrix has a multiplication coefficient in the finite field. How to find the optimal coefficients both keeping the MDS property and further minimizing the implementation area?

In this paper, we aim to overcome the above problems and propose a more general method to construct larger-scale MDS matrices than 4×4 ones towards lower area based on matrix decomposition, automatic search and symbolic computation techniques.

1.2 Contributions

In this paper, we propose a new framework to construct MDS matrices. The key idea is to construct MDS matrices with as few Type-3 elementary matrices as possible first, then minimize the area by finding optimal multiplication coefficients in Type-3 elementary matrices and lastly further reduce the area by introducing a few Type-2 elementary matrices. The main contributions are as follows:

Propose a greedy algorithm to construct MDS matrix patterns by Type-3 elementary matrices. Inspired by matrix decomposition technique, we propose a greedy strategy to construct MDS matrix patterns by a product of Type-3 elementary matrices based on symbolic computation. It is guided by the number of singular submatrices in the resulting matrix after each Type-3 matrix joined. Compared to exhaustive search, our algorithm can significantly reduce the search space so as to improve the efficiency of MDS matrix construction. As applications, we construct 4×4 MDS matrix patterns with 8 Type-3 matrices (vs. previous 8 ones), 5×5 MDS matrix patterns with 13 Type-3 matrices (vs. previous 15 ones), 6×6 MDS matrix patterns with 17 Type-3 matrices (vs. previous 18 ones), 7×7 MDS matrix patterns with 23 Type-3 matrices (vs. previous 28 ones), as well as 8×8 MDS matrix patterns with 26 Type-3 matrices (vs. previous 28 ones).

Propose an automatic method to minimize the area of MDS matrix patterns based on SAT/SMT. To minimize the implementation area of Type-3 matrices once an MDS matrix pattern is obtained in finite field $\mathbb{F}_{2^n}$, we transform the searching of optimal multiplication coefficients in these Type-3 matrices into an optimization problem over multiplicative group $\mathbb{F}_{2^n}^*$, which is also a cyclic group with at least one generator. Based on this theoretical foundation, we incorporate the constraint that all submatrices S of the target MDS matrix pattern must satisfy $\det(S) \neq 0$ into a SAT/SMT-based solving model. This automatic search method can effectively minimize the implementation area of a given MDS matrix pattern.

Propose another greedy strategy to further reduce implementation area by using Type-2 elementary matrices. In fact, adding Type-2 matrices to one side of the sequence of Type-3 matrices does not affect the MDS property, while Type-2 and Type-3 matrices can be exchanged under certain conditions. Based on these findings, we propose another greedy strategy guided by the total implementation cost of Type-2 and Type-3 coefficients. This new strategy can further reduce the implementation area of a given MDS matrix pattern by carefully selecting Type-2 matrices and inserting them into suitable positions of existing decomposition sequences.

Applications. We apply our proposed framework to construct $m \times m$ MDS matrices over $\mathbb{F}_{2^n}$ or $GL(n, \mathbb{F}_2)$, where $m \in \{4, 5, 6, 7, 8\}$ and $n \in \{4, 8, 16, 32, 64\}$. All results are summarized in Table 1. From this table, we can find that our method has several advantages. Firstly, our method is more general and it can be used to construct $m \times m$ MDS matrices where $m \in \{4, 5, 6, 7, 8\}$. Then, the 4×4 MDS matrices constructed by our method can also reach the minimum area, without the need for exhaustive search. The 5×5, 6×6 and 7×7 MDS matrices constructed by our method have lower area compared with previous methods. Since the 8×8 MDS matrices constructed by our method need $26n + 8\texttt{\#XOR}(L) + 2\texttt{\#XOR}(L^{-1}) + 4\texttt{\#XOR}(L^2)$, but EGFS construction needs $28n + 4\texttt{\#XOR}(L) + 4\texttt{\#XOR}(L^{-1}) + 4\texttt{\#XOR}(L^3)$ when $n \in \{16, 32, 64\}$, it easy to find that the matrices constructed by our method have lower area. Unfortunately, when $n = 8$, we do not find a lower-area 8×8 MDS matrix than the ones produced by the EGFS construction.

All experiments were conducted on a desktop equipped with an Intel i7-14700 CPU and 32GB of RAM. The source codes and search results are available at https://github.com/yaaannn/mds-construction/.

1.3 Outline

In Section 2, some preliminaries including MDS matrices theory, matrix decomposition and XOR count of matrices are introduced. In Section 3, a new MDS matrix construction method is presented, including constructing MDS matrix patterns over the symbolic ring, minimizing the implementation area of multiplication coefficients in Type-3 elementary

Table 1: Comparison with previous methods for $m \times m$ MDS matrices with n-bits word size in terms of area cost.

$m \times m$	Method	Cost(#XOR)					Source
		$n=4$	$n=8$	$n=16$	$n=32$	$n=64$	
4	LFSR	68	116	212	404	788	[WWW13]
	GFS	40	72	136	264	520	[WWW13]
	DSI	40	72	-	-	-	[TTKS18]
	DLS	40	72	-	-	-	[GPS22]
	Exhaustive	35	67	-	-	-	[DL18]
	Exhaustive	35	67	-	-	-	[WLTZ23]
	EGFS	35	67	131*	259*	515*	[SM21]
	Ours	**35**	**67**	**131**	**259**	**515**	**Sect. 4.1**
5	LFSR	95	175	335	655	1295	[WWW13]
	DSI	-	155	-	-	-	[TTKS18]
	DSI	-	150	-	-	-	[LSS$^+$19]
	DLS	130	140	-	-	-	[GPS22]
	Ours	**65**	**117**	**215**	**423**	**839**	**Sect. 4.2**
6	LFSR	150	270	510	990	1950	[WWW13]
	GFS	-	216	360	612	1188	[WWW13]
	DSI	-	186	-	-	-	[TTKS18]
	DLS	-	180	-	-	-	[GPS22]
	EGFS	90	156	294*	582*	1158*	[SM21]
	Ours	-	**154**	**290**	**553**	**1097**	**Sect. 4.3**
7	LFSR	210	378	714	1386	2730	[WWW13]
	DSI	-	378	-	-	-	[TTKS18]
	DSI	-	329	-	-	-	[KSV19]
	DLS	-	315	-	-	-	[GPS22]
	Ours	-	**227**	**414**	**759**	**1459**	**Sect. 4.4**
8	LFSR	296	520	968	1864	3656	[WWW13]
	GFS	-	-	640	1152	2112	[WWW13]
	EGFS	-	260	488*	916*	1812*	[SM21]
	Ours	-	**281**	**452**	**850**	**1682**	**Sect. 4.5**

* These results by EGFS construction with $n \in \{16, 32, 64\}$ are calculated and checked by ourselves based on the original paper [SM21].

matrices by an automatic search method, and further reducing the implementation area by Type-2 matrices. As applications, $m \times m$ MDS matrices over $\mathbb{F}_{2^n}$ or $GL(n, \mathbb{F}_2)$ are given in Section 4. Lastly, we conclude this paper in Section 5.

2 Preliminaries

In this section, we briefly recall some concepts and definitions about MDS matrices which will be used throughout this paper.

2.1 MDS Matrices

Let $\mathbb{F}_{2^n}$ be a finite field with 2^n elements and $\mathbb{F}_{2^n}^m$ be an m-dimensional vector space over $\mathbb{F}_{2^n}$, where m and n are positive integers. Indeed, for any linear mapping λ over $\mathbb{F}_{2^n}^m$, there exists an $m \times m$ matrix M over $\mathbb{F}_{2^n}$ such that $\lambda(\mathbf{v}) = M \cdot \mathbf{v}$. Hereafter, we can represent a linear diffusion layer by a linear diffusion matrix of size $m \times m$ over word size n.

Given a vector $\mathbf{v} = (v_0, v_1, \cdots, v_{m-1})^\top \in \mathbb{F}_{2^n}^m$, its *bundle weight* is defined as the number of non-zero components in $\mathbf{v}$, denoted as $\omega(\mathbf{v})$. The branch number of a linear diffusion matrix can be defined as follows.

Definition 1 (Branch number [DR02])**.** Let M be an $m \times m$ matrix over $\mathbb{F}_{2^n}$. Then the differential branch number of M is defined as

$$\mathcal{B}_d(M) = \min_{\mathbf{v} \neq 0} \{\omega(\mathbf{v}) + \omega(M \cdot \mathbf{v})\},$$

while the linear branch number of M is defined as

$$\mathcal{B}_l(M) = \min_{\mathbf{v} \neq 0} \{\omega(\mathbf{v}) + \omega(M^T \cdot \mathbf{v})\}.$$

The branch number usually is used to measure the diffusion property of linear matrices. It is important to note that the maximum value of both $\mathcal{B}_d(M)$ and $\mathcal{B}_l(M)$ is $m + 1$.

When matrix M's differential branch number and linear branch number reach the maximum, M is called an MDS matrix, which is formally defined as follows.

Definition 2 (MDS matrix)**.** Let M be an $m \times m$ matrix over $\mathbb{F}_{2^n}$. Then the matrix M is called an MDS matrix if and only if $\mathcal{B}_d(M) = \mathcal{B}_l(M) = m + 1$.

Beside the above definition, another way to determine whether a matrix is MDS is given by the following lemma, which we also adopt in our work.

Lemma 1. *Let M be an $m \times m$ matrix over $\mathbb{F}_{2^n}$. Then the matrix M is an MDS matrix if and only if every square submatrix of matrix M is non-singular.*

2.2 Matrix Decomposition

Definition 3 (Elementary matrices)**.** Let $\mathbb{F}_{2^n}$ be a finite field and m a positive integer. Elementary matrices over $\mathbb{F}_{2^n}$ fall into three types (or sets):

- **Type-1 (set $\mathbb{S}_1$)**: Row permutation matrices $E_{i,j}$, which interchange the i-th and j-th rows for $i \neq j$.
- **Type-2 (set $\mathbb{S}_2$)**: Row scaling matrices $E_i(\alpha)$, which multiply the i-th row by $\alpha \in \mathbb{F}_{2^n}^*$.
- **Type-3 (set $\mathbb{S}_3$)**: Row addition matrices $E_{i,j}(\beta)$, which add $\beta \in \mathbb{F}_{2^n}$ times the j-th row to the i-th row for $i \neq j$.

To describe these above three types of elementary matrices clearly, we rewrite them in matrix form, which is shown in Appendix A. Actually, elementary matrices are very important in the matrix theory. Every non-singular linear matrix can be decomposed as a product of elementary matrices, as shown in Lemma 1.

Theorem 1 (Matrix decomposition)**.** *Let M be an $m \times m$ invertible matrix over $\mathbb{F}_{2^n}$. Then the matrix M can be decomposed as a product of elementary matrices:*

$$M = G_1 \cdot G_2 \cdot \cdots \cdot G_t, \tag{1}$$

where each $G_i \in \mathbb{S}_1 \cup \mathbb{S}_2 \cup \mathbb{S}_3$ for $i = 1, 2, \ldots, t$.

In [VKS22], some commutative properties are also raised to reorder the elementary matrices from $\mathbb{S}_1, \mathbb{S}_2$ and $\mathbb{S}_3$ in a decomposition, which are illustrated in the following Proposition 1 and Proposition 2.

Proposition 1 ([VKS22])**.** *Let* $f_{i,j}(x) = \begin{cases} i, & \text{if } x = j, \\ j, & \text{if } x = i, \\ x, & \text{otherwise.} \end{cases}$ *Then we have*

(1) $E_{i,j} \cdot E_k(\alpha) = E_{f_{i,j}(k)}(\alpha) \cdot E_{i,j}$,

(2) $E_{i,j} \cdot E_{k,l}(\beta) = E_{f_{i,j}(k), f_{i,j}(l)}(\beta) \cdot E_{i,j}$.

By using Proposition 1, all Type 1 matrices in $\mathbb{S}_1$ can always be moved to the right or left of the sequence in Equation (1). So this equation can be represented equivalently in the following form:

$$M = \underbrace{G_1^1 \cdot G_2^1 \cdots G_{t_1}^1}_{\mathbb{S}_1} \cdot G_1^{'} \cdots G_{t-t_1}^{'}, \tag{2}$$

where $G_i^1 \in \mathbb{S}_1$ for $i = 1, 2, \ldots, t_1$ and $G_j' \in \mathbb{S}_2 \cup \mathbb{S}_3$ for $j = 1, 2, \ldots, t - t_1$.

Similarly, the product of elementary matrices from $\mathbb{S}_2$ and $\mathbb{S}_3$ also satisfies certain commutative properties, which allow their order to be interchanged. These properties are summarized in the following proposition.

Proposition 2 ([VKS22])**.** *Let* $\alpha, \beta \in \mathbb{F}_{2^n}^*$*. Then we have*

$$E_i(\alpha) \cdot E_{j,k}(\beta) = \begin{cases} E_{j,k}(\beta) \cdot E_i(\alpha), & \text{if } i \notin \{j, k\}, \\ E_{j,k}(\alpha\beta) \cdot E_i(\alpha), & \text{if } i = j, \\ E_{j,k}(\alpha^{-1}\beta) \cdot E_i(\alpha), & \text{if } i = k. \end{cases}$$

By using Proposition 2, all Type-2 matrices within the sequence in Equation (2) can be further grouped together. That is

$$M = \underbrace{G_1^1 \cdot G_2^1 \cdots G_{t_1}^1}_{\mathbb{S}_1} \cdot \underbrace{G_1^2 \cdot G_2^2 \cdots G_{t_2}^2}_{\mathbb{S}_2} \cdot \underbrace{G_1^3 \cdot G_2^3 \cdots G_{t_3}^3}_{\mathbb{S}_3}, \tag{3}$$

where $G_i^j \in \mathbb{S}_j$ for $j = 1, 2, 3$. This structured decomposition sequence is useful for both theoretical analysis and efficient implementation of matrices.

2.3 XOR Count

The hardware implementation cost of a linear diffusion layer (i.e., matrix) is usually measured in terms of area and latency. Since only XOR operations are used in linear diffusion matrices, the number of XOR operations is considered as a mean indicator to measure a matrix's implementation area.

There are three methods to count the XOR operations according to different hardware implementations. The first counting method is called direct XOR (d-XOR) count, as shown in Definition 4.

Definition 4 (d-XOR count [KPPY14])**.** The d-XOR count of an $m \times m$ matrix $M \in GL(n, \mathbb{F}_2)$, denoted as d-XOR$(M)$, is determined by

$$\text{d-XOR}(M) = \omega(M) - m,$$

where $\omega(M)$ is the number of nonzero entries in M.

The second counting method is called sequential XOR (s-XOR) count, which is exactly suitable for the implementation of matrices with decomposition. It is defined as follows.

Definition 5 (s-XOR count [JPST17])**.** The s-XOR count of an $m \times m$ matrix $M \in GL(n, \mathbb{F}_2)$, denoted as s-XOR($M$), is represented as

$$M = P \prod_{k=1}^{t} (I + E_{i_k, j_k}),$$

where P is a permutation matrix and each E_{i_k, j_k} ($i_k \neq j_k$, $k = 1, 2, \dots, t$) is a binary matrix with 1 as the (i_k, j_k)-th entry and 0 elsewhere. It is clear that the s-XOR count of M is t.

To show the difference between d-XOR count and s-XOR count, we take the following small 4×4 matrix over $\mathbb{F}_2$ as an example:

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

According to Definition 4, to calculate $(y_1, y_2, y_3, y_4)^T = M \cdot (x_1, x_2, x_3, x_4)^T$, we have

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_1 + x_2 \\ x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 + x_4 \end{bmatrix}.$$

Clearly, the d-XOR count of M is 6. However, based on matrix decomposition theory, we have

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

Thus, the s-XOR count of M is 3 due to Definition 5.

It is easy to find that usually the s-XOR count is smaller than the d-XOR count for the same linear diffusion matrix. Therefore, constructing linear diffusion matrices towards low area with matrix decomposition technique is a good choice.

Since MDS matrices are usually constructed in finite field $\mathbb{F}_{2^n}$, the multiplication of $\alpha \in \mathbb{F}_{2^n}$ given by $x \mapsto \alpha x$ is a linear function over $\mathbb{F}_2$. It should be taken into account when calculating the cost of MDS matrices. Regarding the multiplication $x \mapsto \alpha x$, Beierle et al. obtained the following results.

Lemma 2 ([BKL16])**.** *Let $\alpha \in \mathbb{F}_{2^n}$, then we have*

(1) #XOR$(\alpha) = 0$ if and only if $\alpha = 1$.

(2) #XOR$(\alpha) =$ #XOR(α^{-1}).

(3) #XOR$(\alpha^{\pm k}) \leq k \cdot$ #XOR(α) for $k \geq 1$.

where the XOR count of $\alpha \in \mathbb{F}_{2^n}$ is denoted as #XOR(α). It can be either d-XOR count or s-XOR count, unless specified otherwise.

According to matrix decomposition theory, every linear matrix can be decomposed as a sequence of elementary matrices. Meanwhile, the area cost of this decomposition sequence is determined by Type-2 and Type-3 matrices, as Type-1 matrices do not contribute to implementation area. Thus, we can calculate the cost to implement M as follows.

Definition 6 (Cost of decomposition sequence)**.** Let M be an $m \times m$ matrix over $\mathbb{F}_{2^n}$ with a decomposition sequence according to Equation (3), then the matrix M can be rewritten as

$$M = \underbrace{E_{r_{t_1},s_{t_1}} \cdots E_{r_1,s_1}}_{\mathbb{S}_1} \cdot \underbrace{E_{k_{t_2}}(\beta_{t_2}) \cdots E_{k_1}(\beta_1)}_{\mathbb{S}_2} \cdot \underbrace{E_{i_{t_3},j_{t_3}}(\alpha_{t_3}) \cdots E_{i_1,j_1}(\alpha_1)}_{\mathbb{S}_3}.$$

Then, the cost to implement M can be calculated as

$$\#\texttt{XOR}(M) = t_3 \cdot n + \sum_{i=1}^{t_3} \#\texttt{XOR}(\alpha_i) + \sum_{j=1}^{t_2} \#\texttt{XOR}(\beta_j), \tag{4}$$

where t_2, t_3 are the number of Type-2 and Type-3 matrices, respectively.

Other Notations. For simplicity, we use nonzero positions in each row of a binary matrix as a representation of the matrix. For example, $[[1,2,3],[1,3],[2]]$ represents the binary matrix $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$. We also denote the $m \times m$ identity matrix as I_m.

3 General MDS Matrix Construction Method Towards Low Area

Inspired by matrix decomposition technique for optimizing the implementation of existing matrices, we would like to adopt this technique to construct new low-area MDS matrices.

According to Equation (3), any partitioned matrix can be decomposed into a sequence of elementary matrices. Since Type-1 matrices do not contribute to the branch number and implementation area of the resulting matrix, while Type-2 matrices do not affect the branch number but can reduce the implementation area, we first use a sequence of Type-3 matrices to construct an MDS matrix, then choose suitable Type-2 matrices to further reduce its implementation area. In our construction method, Type-1 matrices are not necessarily needed. Indeed, this is a good approach for constructing low-area MDS matrices. However, there are three main problems:

(1) It is infeasible to exhaust all possible Type-3 matrices as the dimension m increases, because the time complexity is about $O((m^2 - m)^t)$, where t is the number of Type-3 matrices. How to minimize the number of Type-3 matrices so as to reduce the implementation area?

(2) In each Type-3 matrix of the form $E_{i,j}(\alpha)$, there are 2^n possible values of the coefficient $\alpha \in \mathbb{F}_{2^n}$. It is also infeasible to exhaust all possible combinations of $(\alpha_1, \ldots, \alpha_t)$, even if the positions of the Type-3 matrices $(i_1, j_1), (i_2, j_2), \ldots, (i_t, j_t)$ are fixed or partially determined. How to find the optimal coefficients that both satisfy the MDS property and minimize the implementation area?

(3) Since Type-2 matrices can reduce the implementation area of the resulting MDS matrix, how to effectively select and insert the Type-2 matrices within the sequence of Type-3 matrices to further reduce the implementation area?

To solve the above problems, we adopt a *matrix pattern construction – pattern optimization – instantiation* strategy to construct MDS matrices. In Section 3.1, we propose a greedy algorithm to construct MDS matrix patterns by symbolic computation, which can minimize the number of used Type-3 matrices. In Section 3.2, we propose an automatic search model for $(\alpha_1, \ldots, \alpha_t)$ over a fixed finite field based on SAT/SMT to minimize the implementation area as much as possible. In Section 3.3, we propose a greedy strategy that further reduces the implementation area with the help of Type-2 matrices. Combining these three parts together, we are able to search lower-area $m \times m$ MDS matrices, where $m \in \{4, 5, 6, 7, 8\}$.

3.1 Constructing MDS Matrix Patterns by Type-3 Matrices Over Symbolic Ring

According to Equation (3), we can construct an $m \times m$ partitioned MDS matrix M over the finite field $\mathbb{F}_{2^n}$ as a product of Type-3 elementary matrices:

$$M = E_{i_t,j_t}(\alpha_t) \cdots E_{i_2,j_2}(\alpha_2) \cdot E_{i_1,j_1}(\alpha_1) \cdot I_m,$$

where $(i_1, j_1), (i_2, j_2), \ldots, (i_t, j_t)$ and $\alpha_1, \alpha_2, \ldots, \alpha_t$ are undetermined variables.

In total, for an given $m \times m$ partitioned matrix, there are $m(m-1)$ Type-3 matrices of the form $E_{i,j}(\alpha)$, if not considering the value of α. Let $\mathcal{E} = \{E_{i,j}(\alpha) \mid 0 \leq i < m,\ 0 \leq j < m\}$ denote the set of such symbolic Type-3 matrices. Our first goal is to determine the value of $(i_1, j_1), (i_2, j_2), \ldots, (i_t, j_t)$ through symbolic computation so as to minimize the number of Type-3 matrices.

To achieve this goal, we adopt a greedy strategy guided by the number of singular submatrices in the resulting matrix after each Type-3 matrix joined. We initialize the construction with $M_0 = I_m$. At step k for $1 \leq k \leq t$, we select a Type-3 matrix $E_{i_k,j_k}(\alpha)$ that minimizes the number of singular submatrices in the updated matrix $M_k = E_{i_k,j_k}(\alpha) \cdot M_{k-1}$.

Formally, let $\mathcal{S}(M)$ denote the multiset of all square submatrices of M, where identical submatrices occurring at different positions are counted separately, i.e., multiplicities are preserved. We define

$$\mathrm{Sing}(M) = \{S \in \mathcal{S}(M) \mid \det(S) = 0\},$$

where $\det(S)$ is computed symbolically using a computer algebra system such as `SageMath`[1].

At each selection, we choose a Type-3 matrix according to the rule

$$E_{i_k,j_k}(\alpha) = \arg \min_{E_{i,j}(\alpha) \in \mathcal{E}} |\mathrm{Sing}(E_{i,j}(\alpha) \cdot M_{k-1})| ,$$

where the operator $\arg\min$ denotes a Type-3 matrix in $\mathcal{E}$ that minimizes the number of singular submatrices of the resulting product. Of course, it is possible that there are several matrices achieving the same minimum number of singular submatrices, i.e.,

$$\left|\mathrm{Sing}(E_{i_{k'},j_{k'}}(\alpha) \cdot M_{t-1})\right| = \min_{E_{i,j}(\alpha) \in \mathcal{E}} |\mathrm{Sing}(E_{i_k,j_k}(\alpha) \cdot M_{t-1})| .$$

To avoid local optimality or degeneracy, we suggest picking one matrix randomly from these candidates to continue the process. The process terminates once $\mathrm{Sing}(M) = \emptyset$. At this point, all submatrices of M are non-singular. Now, M is a candidate MDS matrix pattern. The above procedures are summarized in Algorithm 1.

To illustrate the process more clearly, we take the construction of a 4×4 MDS matrix pattern as an example, which is shown in Table 2. Note that for a 4×4 matrix, there are 12 Type-3 matrices in total.

Application on MDS matrix patterns. We apply the proposed greedy algorithm to construct $m \times m$ MDS matrix patterns over the symbolic ring. In this setting, the matrix entries are represented symbolically, and the determinants of submatrices are computed symbolically as well. The proposed algorithm tends to minimize the number of Type-3

[1]SageMath is a free open-source mathematics software system. Available at `https://www.sagemath.org/`.

Algorithm 1: Greedy strategy to construct MDS matrix pattern via Type-3 elementary matrices over symbolic ring

```
   Input: Matrix dimension $m \times m$
   Output: Sequence $\mathcal{O} = [E_{i_1,j_1}(\alpha_1), \ldots, E_{i_t,j_t}(\alpha_t)]$ such that
           $M = E_{i_t,j_t}(\alpha_t) \cdots E_{i_1,j_1}(\alpha_1) \cdot I_m$ is MDS matrix over symbolic ring
 1 Initialize $M \leftarrow I_m$, $\mathcal{O} \leftarrow [\,]$;
 2 while Sing($M$) $\neq \emptyset$ do
 3     Let $\mathcal{E} \leftarrow \{E_{i,j}(\alpha) \mid i \neq j, \alpha$ is symbolic variables$\}$;
 4     Initialize best_set $\leftarrow [\,]$, min_sing $\leftarrow \infty$;
 5     foreach $E_{i,j}(\alpha) \in \mathcal{E}$ do
 6         $M' \leftarrow E_{i,j}(\alpha) \cdot M$;
 7         Compute $s \leftarrow |\,\mathrm{Sing}(M')|$;
 8         if $s < min\_sing$ then
 9             best_set $\leftarrow [E_{i,j}(\alpha)]$;
10             min_sing $\leftarrow s$;
11         else if $s = min\_sing$ then
12             Append $E_{i,j}(\alpha)$ to best_set;
13     Randomly select $E^{\star}_{i,j}(\alpha) \in$ best_set;
14     $M \leftarrow E^{\star}_{i,j}(\alpha) \cdot M$;
15     Append $E^{\star}_{i,j}(\alpha)$ to $\mathcal{O}$;
16 return $\mathcal{O}$;
```

elementary operations required to reach an MDS matrix. Unlike exhaustive enumeration or algebraic constructions, this approach yields highly compact decompositions, which is beneficial for searching MDS matrices with low implementation area.

Table 3 summarizes the number of Type-3 operations required to construct symbolic MDS matrix patterns of different dimensions. Our results are compared with existing constructions reported in the literature, which shows that our method often achieves shorter decomposition lengths, especially for MDS matrices with higher dimensions.

3.2 Minimize the Implementation Area of Multiplication Coefficients in Type-3 Matrices

Once an MDS matrix pattern is constructed by a sequence of Type-3 elementary matrices, it can be written as:

$$M = E_{i_t,j_t}(\alpha_t) \cdots E_{i_2,j_2}(\alpha_2) \cdot E_{i_1,j_1}(\alpha_1).$$

Here, the positions $(i_1, j_1), \ldots, (i_t, j_t)$ are fixed. The next step is to determine the values of $\alpha_1, \ldots, \alpha_t$ in order to minimize the total XOR cost $\sum_{i=1}^{t} \texttt{\#XOR}(\alpha_i)$, thereby optimizing the implementation area while ensuring that the resulting matrix M remains MDS property.

We focus on constructing MDS matrices over the finite field $\mathbb{F}_{2^n}$, where the multiplicative group $\mathbb{F}_{2^n}^*$ is a cyclic group with order 2^n-1. Let g be a generator of this group, then for any $\alpha \in \mathbb{F}_{2^n}^*$, there exists an integer $\lambda \in [0, 2^n-2]$ such that $\alpha = g^{\lambda}$.

Thus, the matrix can be rewritten as:

$$M = E_{i_t,j_t}(g^{\lambda_t}) \cdots E_{i_1,j_1}(g^{\lambda_1}), \quad \text{with } 0 \leq \lambda_i < 2^n-1.$$

Finding the optimal tuple $(\alpha_1, \ldots, \alpha_t)$ is equivalent to finding $(\lambda_1, \ldots, \lambda_t)$ such that the total cost $\sum_{i=1}^{t} \texttt{\#XOR}(g^{\lambda_i})$ is minimized. Meanwhile, for each square submatrix $S \subseteq M$,

Table 2: An example of greedy strategy to construct a 4×4 MDS matrix pattern

Step	Selections of Type-3 matrix	MDS matrix pattern	$\lvert\mathrm{Sing}(M)\rvert$
0	—	$M_0 = I_4$	54
1	All 12 Type-3 matrices	$M_1 = E_{0,1}(\alpha_1) \cdot M_0$	50
2	$\{E_{2,3}(\alpha_2), E_{3,2}(\alpha_2)\}$	$M_2 = E_{2,3}(\alpha_2) \cdot M_1$	45
3	$\{E_{1,2}(\alpha_3), E_{3,0}(\alpha_3)\}$	$M_3 = E_{1,2}(\alpha_3) \cdot M_2$	36
4	$\{E_{3,0}(\alpha_4)\}$	$M_4 = E_{3,0}(\alpha_4) \cdot M_3$	24
5	$\{E_{0,1}(\alpha_5), E_{2,3}(\alpha_5)\}$	$M_5 = E_{0,1}(\alpha_5) \cdot M_4$	16
6	$\{E_{2,3}(\alpha_6)\}$	$M_6 = E_{2,3}(\alpha_6) \cdot M_5$	6
7	$\{E_{1,2}(\alpha_7), E_{3,0}(\alpha_7)\}$	$M_7 = E_{1,2}(\alpha_7) \cdot M_6$	2
8	$\{E_{3,0}(\alpha_8)\}$	$M_8 = E_{3,0}(\alpha_8) \cdot M_7$	0

Table 3: Number of Type-3 elementary matrices required to construct MDS matrices

$m \times m$	LFSR [WWW13]	DSI [KSV19]	DLS [GPS22]	GFS [SM21]	This Work
4×4	12	8	8	8	**8**
5×5	20	15	15	-	**13**
6×6	30	18	18	18	**17**
7×7	42	28	28	-	**23**
8×8	56	-	-	28	**26**

we symbolically compute its determinant:

$$\det(S) = g^{a_1} + g^{a_2} + \cdots + g^{a_s} \neq 0,$$

where the exponents $a_1, \ldots, a_s$ are linear combinations of $\lambda_1, \ldots, \lambda_t$. This condition ensures the non-singularity of all submatrices, i.e., the MDS property.

Given the pattern, the exponent assignment problem can be formulated as the following optimization problem:

$$\begin{cases} \min \sum_{i=1}^{t} \texttt{\#XOR}(g^{\lambda_i}), \\ \det(S) = g^{a_1} + \cdots + g^{a_s} \neq 0, \quad \forall S \subseteq M. \end{cases}$$

To solve this problem, we propose an automatic search model based on SAT/SMT. Since $\mathbb{F}_{2^n}^*$ is a cyclic group, we have:

$$g^{\lambda} = g^{\lambda \bmod (2^n-1)}, \quad \text{so} \quad g^{-1} = g^{2^n-2}, \quad g^{-2} = g^{2^n-3}, \ldots$$

Usually, the implementation area cost of g^{λ} is smaller when $|\lambda|$ is smaller. Therefore, we approximate:

$$\texttt{\#XOR}(g^{\lambda_1}) \leq \texttt{\#XOR}(g^{\lambda_2}) \quad \text{if } |\lambda_1| < |\lambda_2|, \quad \lambda_i \in \left[-(2^{n-1}-1),\, 2^{n-1}-1\right].$$

Hence, the minimization target $\sum_{i=1}^{t} \texttt{\#XOR}(g^{\lambda_i})$ can be approximated by $\sum_{i=1}^{t} |\lambda_i|$. Likewise, the constraint

$$\det(S) = g^{a_1} + \cdots + g^{a_s} \neq 0$$

can be rewritten as

$$\det(S) = g^{a_1 \bmod (2^n-1)} + \cdots + g^{a_s \bmod (2^n-1)} \neq 0.$$

However, it is difficult to directly determine whether $\det(S) \neq 0$ holds in $\mathbb{F}_{2^n}$ when using this primitive representation. To address this challenge, we transform it into a polynomial representation. Specifically, we define a mapping:

$$\phi(\lambda) : \lambda \mapsto g^{\lambda} \bmod f(x),$$

where $f(x) \in \mathbb{F}_2[x]$ is a fixed irreducible polynomial defining $\mathbb{F}_{2^n}$. The mapping ϕ can be stored as a pre-computed table indexed by λ, mapping each exponent to its polynomial representation in vector form. For instance, in $\mathbb{F}_{2^4}$ with $f(x) = x^4 + x + 1$, an example of the mapping ϕ is shown in Table 4.

Table 4: Pre-computed table for $\phi(\lambda) = g^\lambda \bmod f(x)$ in $\mathbb{F}_{2^4}$ with $f(x) = x^4 + x + 1$

λ	$\phi(\lambda)$ by polynomial representation	$\phi(\lambda)$ by vector representation
0	1	(0,0,0,1)
1	g^1	(0,0,1,0)
2	g^2	(0,1,0,0)
3	g^3	(1,0,0,0)
4	$g+1$	(0,0,1,1)
5	g^2+g	(0,1,1,0)
6	g^3+g^2	(1,1,0,0)
7	g^3+g+1	(1,0,1,1)
8	g^2+1	(0,1,0,1)
9	g^3+g	(1,0,1,0)
10	g^2+g+1	(0,1,1,1)
11	g^3+g^2+g	(1,1,1,0)
12	g^3+g^2+g+1	(1,1,1,1)
13	g^3+g^2+1	(1,1,0,1)
14	g^3+1	(1,0,0,1)

With the help of the pre-computed table ϕ, the determinant constraint is rewritten as:

$$\det(S) = \phi(a_1 \bmod 2^n{-}1) \oplus \cdots \oplus \phi(a_s \bmod 2^n{-}1) \neq 0,$$

where $\oplus$ denotes bitwise XOR in $\mathbb{F}_{2^n}$, replacing addition in the exponent domain.

In summary, the optimization problem is reformulated as:

$$\begin{cases} \min \sum_{i=1}^{t} |\lambda_i|, \\ -(2^{n-1}{-}1) \leq \lambda_i \leq 2^{n-1}{-}1, \quad i = 1, 2, \ldots, t \\ \det(S) = \phi(a_1 \bmod 2^n{-}1) \oplus \cdots \oplus \phi(a_s \bmod 2^n{-}1) \neq 0, \quad \forall S \subseteq M. \end{cases}$$

This formulation can be solved by SMT solvers (e.g., Z3[2]). In practice, we may incorporate an additional constraint $\sum_{i=1}^{t} |\lambda_i| \leq C_{\max}$, or perform optimization across all satisfying assignments. This approach effectively guides the selection of exponent assignments with lower implementation area, while ensuring the MDS property. The whole process is summarized in Algorithm 2.

3.3 Further Reduce the Implementation Area by Type-2 Matrices

At this point, an MDS matrix pattern can be constructed by a sequence of Type-3 matrices as:

$$M = E_{i_t,j_t}(g^{\lambda_t}) \cdots E_{i_2,j_2}(g^{\lambda_2}) \cdot E_{i_1,j_1}(g^{\lambda_1}),$$

where the positions $(i_1, j_1), \ldots, (i_t, j_t)$ and the exponents $\lambda_1, \ldots, \lambda_t$ are fixed.

Next, we aim to introduce Type-2 matrices into the sequence to further reduce the implementation area. Since inserting Type-2 matrices at the left or the right of the sequence does not affect the MDS property, we can add them individually at either end of the sequence.

[2]Z3 is a high-performance SMT solver. Available at https://microsoft.github.io/z3guide/.

Algorithm 2: SMT-based automatic model to assign exponents of multiplication coefficients in Type-3 matrices

```
Input: Symbolic matrix $M$ with symbolic exponents $\lambda_1, \dots, \lambda_t$, irreducible
       polynomial $f(x) \in \mathbb{F}_2[x]$, finite field $\mathbb{F}_{2^n}$
Output: Assignment $(\lambda_1, \dots, \lambda_t) \in \mathbb{Z}^t$ such that determinants of all square
        submatrices of $M$ are nonzero in $\mathbb{F}_{2^n}$
// Step 1: Initialize SMT environment
1  Declare $t$ bitvector variables $\lambda_1, \dots, \lambda_t \in \texttt{BitVec}(k)$;
2  Constrain each $\lambda_i$ to represent an integer in $\left[-(2^{n-1}-1),\ 2^{n-1}-1\right]$;
3  Constrain $\sum_{i=1}^{t} |\lambda_i| \le C_{\max}$ ;            // Optional upper bound
// Step 2: Define pre-computed table
4  Define mapping $\phi(\lambda) : \lambda \mapsto g^{\lambda} \bmod f(x)$;
5  Implement $\phi$ as a pre-computed over all possible $\lambda \bmod (2^n-1)$;
// Step 3: Encode determinant constraints
6  foreach square submatrix $S \subseteq M$ do
7      Compute symbolic determinant: $\det(S) = L^{a_1} + L^{a_2} + \cdots + L^{a_s}$;
8      Let $v_1 = \phi(a_1), \dots, v_s = \phi(a_s)$;
9      Add constraint: $v = v_1 \oplus v_2 \oplus \cdots \oplus v_m \neq \mathbf{0}$;
// Step 4: Solve and extract solution
10 Write SMT constraints to a file and invoke an SMT solver.;
11 if solution exists then
12     return model $(\lambda_1, \dots, \lambda_t)$
13 else
14     return unsatisfiable — no valid instantiation found
```

According to Proposition (3), Type-2 and Type-3 matrices can be exchanged under specific transformation rules as follows:

$$E_{i,j}(g^{\lambda}) \cdot E_k(g^{\gamma}) = \begin{cases} E_i(g^c) \cdot E_{i,j}(g^{\lambda-\gamma}) & \text{if } k = i, \\ E_i(g^c) \cdot E_{i,j}(g^{\lambda+\gamma}) & \text{if } k = j, \\ E_i(g^c) \cdot E_{i,j}(g^{\lambda}) & \text{otherwise.} \end{cases}$$

Moreover, two Type-2 matrices can also be exchanged according to the following rule:

$$E_k(g^{\gamma}) \cdot E_l(g^{\delta}) = \begin{cases} E_k(g^{\gamma+\delta}) & \text{if } k = l, \\ E_k(g^{\gamma}) \cdot E_l(g^{\delta}) & \text{otherwise.} \end{cases}$$

Thus, each Type-2 matrix $E_k(g^{\gamma})$ can be propagated from the right to the left of the sequence, during which the exponents λ_i in the Type-3 matrices may change accordingly. By carefully selecting suitable Type-2 matrices and inserting them at appropriate positions in the sequence, the total XOR cost can be further reduced.

To achieve it further, we adopt a greedy strategy. In total, there are $n\cdot(2^n-1)$ candidate Type-2 matrices of the form $E_k(g^r)$, where $0 \le k < n$ and $\gamma \in \left[-(2^{n-1}-1),\ 2^{n-1}-1\right]$, as well as t possible insertion points in the sequence of Type-3 matrices. In order to find the optimal Type-2 matrix and insertion position, we perform the following steps:

Step 1: Propagate each candidate Type-2 matrix to all t insertion points from the right to the left of the sequence, and compute the updated XOR cost after insertion. This results in evaluating $m(2^n-1)t$ configurations in total, and their costs are recorded.

Step 2: Select the Type-2 matrix and insertion position that minimized total updated XOR cost.

Step 3: If the updated minimum cost is not lower than the current cost, we terminate the process. Otherwise, we update the matrix sequence by inserting the selected Type-2 matrix at the chosen position, then go back to Step 1.

To describe the above procedure more clearly and highlight the greedy strategy, we summarize it in Algorithm 3.

Algorithm 3: Greedy strategy to reduce the implementation area via Type-2 matrices

```
Input: Type-3 matrix sequence $\mathcal{O} = [E_{i_1,j_1}(g^{\lambda_1}), \dots, E_{i_t,j_t}(g^{\lambda_t})]$
Output: Optimized sequence $\mathcal{O}'$
1  Initialize loop ← true;
2  Initialize OldCost ← $\sum_{j=1}^{t} |\lambda_j|$;
3  while loop do
4      Set loop ← false;
5      Let best_gain ← 0, best_insertion ← None;
       // Try all possible Type-2 insertions
6      foreach row index $k = 0$ to $m-1$ do
7          foreach exponent $\gamma \in \left[-(2^{n-1}-1),\, 2^{n-1}-1\right]$ do
8              Insert $E_k(g^\gamma)$ at the right of $\mathcal{O}$;
               // Propagate $E_k(g^\gamma)$ from right to left
9              for $s = t$ to 1 do
10                 Swap $E_{i_s,j_s}(g^{\lambda_s})$ with $E_k(g^\gamma)$;
11                 Compute new cost NewCost ← $\sum_{j=1}^{t} |\lambda_j| + \sum_{\text{Type-2}} |\gamma|$;
12                 Let $\Delta$ ← OldCost − NewCost;
13                 if $\Delta >$ best_gain then
14                     Update best_gain ← $\Delta$;
15                     Update best_insertion ← $E_k(g^\gamma)$;
16     if best_gain > 0 then
17         Append best_insertion to sequence $\mathcal{O}$;
18         Update OldCost ← NewCost;
19         Set loop ← true;
20 return Optimized operation sequence $\mathcal{O}'$
```

Discussion. Usually, we construct MDS matrices by using Type-2 and Type-3 elementary matrices over finite fields $\mathbb{F}_{2^4}$ and $\mathbb{F}_{2^8}$. These matrices are efficiently used for designing traditional block ciphers. However, post-quantum secure cryptographic primitives require linear layers over larger word sizes, such as 16, 32 and 64 bits. Fortunately, our method can also construct MDS matrices over such large word sizes by replacing the generator $g \in \mathbb{F}_{2^n}$ used in the matrix decomposition with a binary matrix $L \in GL(n, \mathbb{F}_2)$. This generalization is motivated by the fact that every generator g can be represented as an $n \times n$ binary matrix $L \in GL(n, \mathbb{F}_2)$ under a fixed basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$. In order to control the implementation area, the XOR cost of L is usually limited to several XORs. In this work, we fix a special structure for the matrix L as

$$L[i+1, i] = 1 \quad \text{for } 0 \leq i < n-1, \qquad L[0, n-1] = 1.$$

Another $1 \sim 3$ extra '1's are randomly placed in the remained entries of the matrix. The construction results are shown in the next section.

4 Applications

4.1 Construction of 4×4 MDS Matrices

Firstly, according to Algorithm 1, we can construct potential 4×4 MDS matrix patterns using 8 Type-3 elementary matrices. For example, consider an MDS matrix pattern of the form:

$$M = E_{3,0}(L^{\lambda_8})\cdot E_{1,2}(L^{\lambda_7})\cdot E_{2,3}(L^{\lambda_6})\cdot E_{0,1}(L^{\lambda_5})\cdot E_{3,0}(L^{\lambda_4})\cdot E_{1,2}(L^{\lambda_3})\cdot E_{2,3}(L^{\lambda_2})\cdot E_{0,1}(L^{\lambda_1}),$$

where L and exponents λ_i $(i = 1, 2, \dots, 8)$ are symbolic variables yet to be determined. The cost of this pattern is given by $8n + \sum_{i=1}^{8} \texttt{\#XOR}(L^{\lambda_i})$.

Next, we minimize the sum $\sum_{i=1}^{8} |\lambda_i|$ by automatically searching for suitable values of λ_i based on SAT/SMT, as described in Algorithm 2, over a fixed finite field such as $\mathbb{F}_{2^4}/0\text{x}13$ or $\mathbb{F}_{2^8}/0\text{x}1\text{c}3$.

As a result, we find that one of the best exponent instantiations over $\mathbb{F}_{2^4}/0\text{x}13$ is

$$\begin{cases} \lambda_1 = \lambda_2 = \lambda_3 = \lambda_8 = 0 \\ \lambda_4 = \lambda_5 = \lambda_6 = -1 \\ \lambda_7 = 1 \end{cases},$$

and the resulting matrix M is

$$M = E_{3,0}(1) \cdot E_{1,2}(L) \cdot E_{2,3}(L^{-1}) \cdot E_{0,1}(L^{-1}) \cdot E_{3,0}(L^{-1}) \cdot E_{1,2}(1) \cdot E_{2,3}(1) \cdot E_{0,1}(1).$$

The cost to implement this matrix is $8n + \texttt{\#XOR}(L) + 3\texttt{\#XOR}(L^{-1})$.

Furthermore, we apply Type-2 elementary matrices to optimize M again, according to Algorithm 3. The resulting matrix is:

$$M = E_{3,0}(1)\cdot E_{1,2}(1)\cdot E_{2,3}(L^{-1})\cdot E_{0,1}(1)\cdot E_{1}(L^{-1})\cdot E_{3,0}(L^{-1})\cdot E_{1,2}(1)\cdot E_{2,3}(1)\cdot E_{0,1}(1). \quad (5)$$

The cost to implement M is now approximately $8n + 3\texttt{\#XOR}(L^{-1})$, which is equivalent to the best previous result.

Lastly, we choose an $n \times n$ binary matrix L and verify whether the corresponding matrix M is MDS matrix by computing all its minors and checking that they are non-zero according to Lemma 1. Finally, we successfully obtain 4×4 MDS matrices with word sizes $n = 4, 8, 16, 32$ and 64 bits, as summarized in Table 5.

Table 5: Construction of 4×4 MDS matrices with word sizes $n = 4, 8, 16, 32$ and 64 bits

n	Field/Ring	Pattern	L	#XOR
4	$\mathbb{F}_{2^4}/0x13$	Exp. (5)	$[[4],[1,4],[2],[3]]^*$	35
8	$GL(8,\mathbb{F}_2)$	Exp. (5)	$[[2,8],[1],[2],\dots,[7]]$	67
16	$GL(16,\mathbb{F}_2)$	Exp. (5)	$[[1,16],[1],[2],\dots,[15]]$	131
32	$GL(32,\mathbb{F}_2)$	Exp. (5)	$[[2,32],[1],[2],\dots,[31]]$	259
64	$GL(64,\mathbb{F}_2)$	Exp. (5)	$[[1,64],[1],[2],\dots,[63]]$	515

* L is the companion matrix of $x^4 + x + 1 \in \mathbb{F}_2[x]$.

4.2 Construction of 5×5 MDS Matrices

Firstly, according to Algorithm 1, we can construct potential 5×5 MDS matrix patterns by 13 Type-3 elementary matrices. For example, consider an MDS matrix pattern as follows:

$$\begin{aligned} M = {} & E_{4,3}(L^{\lambda_{13}}) \cdot E_{1,0}(L^{\lambda_{12}}) \cdot E_{2,4}(L^{\lambda_{11}}) \cdot E_{3,1}(L^{\lambda_{10}}) \cdot E_{0,2}(L^{\lambda_9}) \\ & \cdot E_{2,3}(L^{\lambda_8}) \cdot E_{1,0}(L^{\lambda_7}) \cdot E_{4,1}(L^{\lambda_6}) \cdot E_{0,4}(L^{\lambda_5}) \cdot E_{3,0}(L^{\lambda_4}) \\ & \cdot E_{1,2}(L^{\lambda_3}) \cdot E_{2,3}(L^{\lambda_2}) \cdot E_{0,1}(L^{\lambda_1}), \end{aligned}$$

where L and the exponents λ_i $(i = 1, 2, \ldots, 13)$ are symbolic variables and they are unknown. Now the cost of this pattern is $13n + \sum_{i=1}^{13} \texttt{\#XOR}(L^{\lambda_i})$.

Next, we minimize the sum of the exponents $\sum_{i=1}^{13} \lambda_i$ by automatically searching suitable values for λ_i, $i = 1, 2, \ldots, 13$ based on SAT/SMT according to the Algorithm 2 over a fixed finite field, such as $\mathbb{F}_{2^4}/0x13$. As a result, we find that one of the best exponent instantiations over $\mathbb{F}_{2^4}/0x13$ is

$$\begin{cases} \lambda_1 = \lambda_5 = \lambda_7 = \lambda_8 = \lambda_9 = \lambda_{11} = \lambda_{13} = 0 \\ \lambda_3 = -6 \quad \lambda_6 = -3 \quad \lambda_{12} = -1 \\ \lambda_{10} = 1 \quad \lambda_4 = 2 \quad \lambda_2 = 3 \end{cases}.$$

Applying further optimization via Type-2 elementary matrices by Algorithm 3, we obtain the following matrix pattern:

$$\begin{aligned} M = E_{4,3}(1) \cdot E_{1,0}(1) \cdot E_{2,4}(1) \cdot E_{3,1}(1) \cdot E_1(L) \\ \cdot E_{0,2}(1) \cdot E_{2,3}(1) \cdot E_{1,0}(1) \cdot E_{4,1}(L^{-3}) \cdot E_{0,4}(1) \\ \cdot E_{3,0}(L^2) \cdot E_{1,2}(L^{-6}) \cdot E_{2,3}(L^3) \cdot E_{0,1}(1). \end{aligned} \tag{6}$$

Now the cost of this matrix is $13n + \texttt{\#XOR}(L) + \texttt{\#XOR}(L^2) + \texttt{\#XOR}(L^3) + \texttt{\#XOR}(L^{-3}) + \texttt{\#XOR}(L^{-6})$.

In order to construct the MDS matrix with 8-bit word size, we fix the finite field as $\mathbb{F}_{2^8}/0x187$ and apply Algorithm 2 again to find a satisfying assignment for the symbolic exponents:

$$\begin{cases} \lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = \lambda_7 = \lambda_{11} = \lambda_{13} = 0 \\ \lambda_8 = \lambda_9 = \lambda_{10} = \lambda_{12} = -1 \\ \lambda_6 = -3 \end{cases}.$$

Now the matrix M is

$$\begin{aligned} M = & E_{4,3}(1) \cdot E_{1,0}(L^{-1}) \cdot E_{2,4}(1) \cdot E_{3,1}(L^{-1}) \\ & \cdot E_{0,2}(L^{-1}) \cdot E_{2,3}(L^{-1}) \cdot E_{1,0}(1) \cdot E_{4,1}(L^{-3}) \cdot E_{0,4}(1) \\ & \cdot E_{3,0}(1) \cdot E_{1,2}(1) \cdot E_{2,3}(1) \cdot E_{0,1}(1). \end{aligned} \tag{7}$$

The cost of this matrix is $13n + 4\texttt{\#XOR}(L^{-1}) + \texttt{\#XOR}(L^{-3})$.

Based on Expression (6) and (7), we choose an $n \times n$ binary matrix L to check if its corresponding matrix M is MDS matrix by computing the minors of M and checking if they are non-zero according to Lemma 1. Finally, we find 5×5 MDS matrices with word sizes $n = 4, 8, 16, 32$ and 64 bits, which are summarized in Table 6.

Table 6: Construction of 5×5 MDS matrices with word sizes $n = 4, 8, 16, 32$ and 64 bits

n	Field/Ring	Pattern	L	#XOR
4	$\mathbb{F}_{2^4}/0x13$	Exp. (6)	$[[4],[1,4],[2],[3]]^*$	65
8	$\mathbb{F}_{2^8}/0x187$	Exp. (7)	$[[8],[1,2],[2,8],\ldots,[7]]^\star$	117
16	$GL(16,\mathbb{F}_2)$	Exp. (7)	$[[1,16],[1],[2],\ldots,[15]]$	215
32	$GL(32,\mathbb{F}_2)$	Exp. (7)	$[[2,32],[1],[2],\ldots,[31]]$	423
64	$GL(64,\mathbb{F}_2)$	Exp. (7)	$[[1,64],[1],[2],\ldots,[63]]$	839

* L is the companion matrix of $x^4 + x + 1 \in \mathbb{F}_2[x]$.
* L is the companion matrix of $x^8 + x^7 + x^2 + x + 1 \in \mathbb{F}_2[x]$.

4.3 Construction of 6×6 MDS Matrices

Firstly, according to Algorithm 1, we can construct potential 6×6 MDS matrix patterns by 17 Type-3 elementary matrices. For example, consider an MDS matrix pattern as follows:

$$\begin{aligned} M = & E_{3,0}(L^{\lambda_{17}}) \cdot E_{1,4}(L^{\lambda_{16}}) \cdot E_{4,3}(L^{\lambda_{15}}) \cdot E_{2,1}(L^{\lambda_{14}}) \cdot E_{0,5}(L^{\lambda_{13}}) \\ & \cdot E_{5,4}(L^{\lambda_{12}}) \cdot E_{3,2}(L^{\lambda_{11}}) \cdot E_{1,0}(L^{\lambda_{10}}) \cdot E_{4,1}(L^{\lambda_{9}}) \cdot E_{2,5}(L^{\lambda_{8}}) \\ & \cdot E_{0,3}(L^{\lambda_{7}}) \cdot E_{5,0}(L^{\lambda_{6}}) \cdot E_{3,4}(L^{\lambda_{5}}) \cdot E_{1,2}(L^{\lambda_{4}}) \cdot E_{4,5}(L^{\lambda_{3}}) \\ & \cdot E_{2,3}(L^{\lambda_{2}}) \cdot E_{0,1}(L^{\lambda_{1}}), \end{aligned}$$

where L and exponents λ_i $(i = 1, 2, \ldots, 17)$ are symbolic variables and they are unknown. Now the cost of this pattern is $13n + \sum_{i=1}^{17} \texttt{\#XOR}(L^{\lambda_i})$.

Next, we minimize the sum of the exponents $\sum_{i=1}^{17} |\lambda_i|$ by automatically searching suitable values for λ_i, $i = 1, 2, \ldots, 17$ based on SAT/SMT according to Algorithm 2 over a fixed finite field. However, we cannot find a satisfying assignment for the symbolic exponents over $\mathbb{F}_{2^4}$. Therefore, we try to find a satisfying assignment over $\mathbb{F}_{2^8}$. As a result, we find that one of the best exponent instantiations over $\mathbb{F}_{2^8}/0\text{x}187$ is

$$\begin{cases} \lambda_1 = \lambda_2 = \lambda_3 = \lambda_8 = \lambda_{10} = \lambda_{13} = \lambda_{14} = \lambda_{15} = \lambda_{16} = \lambda_{17} = 0 \\ \lambda_7 = \lambda_{11} = -2 \qquad \lambda_4 = \lambda_5 = \lambda_9 = \lambda_{12} = 1 \\ \lambda_6 = -1 \end{cases}.$$

Now the matrix M is

$$\begin{aligned} M = & E_{3,0}(1) \cdot E_{1,4}(1) \cdot E_{4,3}(1) \cdot E_{2,1}(1) \cdot E_{0,5}(1) \\ & \cdot E_{5,4}(L) \cdot E_{3,2}(L^{-2}) \cdot E_{1,0}(1) \cdot E_{4,1}(L) \cdot E_{2,5}(1) \\ & \cdot E_{0,3}(L^{-2}) \cdot E_{5,0}(L^{-1}) \cdot E_{3,4}(L) \cdot E_{1,2}(L) \cdot E_{4,5}(1) \\ & \cdot E_{2,3}(1) \cdot E_{0,1}(1), \end{aligned} \tag{8}$$

and the cost of this matrix is $17n + 4\texttt{\#XOR}(L) + \texttt{\#XOR}(L^{-1}) + 2\texttt{\#XOR}(L^{-2})$.

Based on Expression (8), we choose an $n \times n$ binary matrix L to check if its corresponding matrix M is MDS by computing the minors of M and checking if they are non-zero according to Lemma 1. Finally, we obtain 6×6 MDS matrices with word sizes $n = 8, 16, 32$ and 64 bits, which are summarized in Table 7.

Table 7: Construction of 6×6 MDS matrices with word sizes $n = 8, 16, 32$ and 64 bits

n	Field/Ring	Pattern	L	#XOR
8	$\mathbb{F}_{2^8}/0\text{x}187$	Exp. (8)	$[[8], [1, 2], [2, 8], \ldots, [7]]^*$	154
16	$GL(16, \mathbb{F}_2)$	Exp. (8)	$[[2, 16], [1], [2, 5], [3], \ldots, [15]]$	290
32	$GL(32, \mathbb{F}_2)$	Exp. (8)	$[[11, 32], [1], [2], \ldots, [31]]$	553
64	$GL(64, \mathbb{F}_2)$	Exp. (8)	$[[13, 64], [1], [2], \ldots, [63]]$	1097

* L is the companion matrix of $x^8 + x^7 + x^2 + x + 1 \in \mathbb{F}_2[x]$.

4.4 Construction of 7 × 7 MDS Matrices

Firstly, according to Algorithm 1, we can construct potential 7×7 MDS matrix patterns by 23 Type-3 elementary matrices. For example, consider an MDS matrix pattern as follows:

$$\begin{aligned} M = {} & E_{0,2}(L^{\lambda_{23}}) \cdot E_{5,1}(L^{\lambda_{22}}) \cdot E_{1,0}(L^{\lambda_{21}}) \cdot E_{0,6}(L^{\lambda_{20}}) \cdot E_{4,1}(L^{\lambda_{19}}) \\ & \cdot E_{6,4}(L^{\lambda_{18}}) \cdot E_{2,6}(L^{\lambda_{17}}) \cdot E_{4,5}(L^{\lambda_{16}}) \cdot E_{3,0}(L^{\lambda_{15}}) \cdot E_{5,2}(L^{\lambda_{14}}) \\ & \cdot E_{1,3}(L^{\lambda_{13}}) \cdot E_{0,4}(L^{\lambda_{12}}) \cdot E_{3,5}(L^{\lambda_{11}}) \cdot E_{4,1}(L^{\lambda_{10}}) \cdot E_{2,0}(L^{\lambda_{9}}) \\ & \cdot E_{6,3}(L^{\lambda_{8}}) \cdot E_{0,6}(L^{\lambda_{7}}) \cdot E_{5,0}(L^{\lambda_{6}}) \cdot E_{3,4}(L^{\lambda_{5}}) \cdot E_{1,2}(L^{\lambda_{4}}) \\ & \cdot E_{4,5}(L^{\lambda_{3}}) \cdot E_{2,3}(L^{\lambda_{2}}) \cdot E_{0,1}(L^{\lambda_{1}}), \end{aligned}$$

where L and exponents λ_i ($i = 1, 2, \ldots, 23$) are symbolic variables and they are unknown. Now the cost of this pattern is $21n + \sum_{i=1}^{23} \#\texttt{XOR}(L^{\lambda_i})$.

Next, we minimize the sum of the exponents $\sum_{i=1}^{23} |\lambda_i|$ by automatically searching suitable values for λ_i, $i = 1, 2, \ldots, 23$ based on SAT/SMT according to Algorithm 2 over a fixed finite field. However, we cannot find a satisfying assignment for the symbolic exponents over $\mathbb{F}_{2^4}$. Therefore, we try to find a satisfying assignment over $\mathbb{F}_{2^8}$. As a result, we find that one of the best exponent instantiations over $\mathbb{F}_{2^8}/0\text{x}187$ is

$$\begin{cases} \lambda_1 = \lambda_2 = \lambda_3 = \lambda_{21} = \lambda_{23} = 0 \\ \lambda_4 = \lambda_{10} = \lambda_{11} = -3 \qquad \lambda_{16} = \lambda_{22} = 1 \\ \lambda_5 = \lambda_{18} = \lambda_{20} = -2 \qquad \lambda_9 = \lambda_{15} = \lambda_{19} = 2 \\ \lambda_6 = \lambda_7 = \lambda_8 = \lambda_{12} = \lambda_{13} = \lambda_{14} = \lambda_{17} = -1 \end{cases}.$$

Now the matrix M is

$$\begin{aligned} M = {} & E_{0,2}(1) \cdot E_{5,1}(L) \cdot E_{1,0}(1) \cdot E_{0,6}(L^{-2}) \cdot E_{4,1}(L^{2}) \\ & \cdot E_{6,4}(L^{-2}) \cdot E_{2,6}(L^{-1}) \cdot E_{4,5}(L) \cdot E_{3,0}(L^{2}) \cdot E_{5,2}(L^{-1}) \\ & \cdot E_{1,3}(L^{-1}) \cdot E_{0,4}(L^{-1}) \cdot E_{3,5}(L^{-3}) \cdot E_{4,1}(L^{-3}) \cdot E_{2,0}(L^{2}) \\ & \cdot E_{6,3}(L^{-1}) \cdot E_{0,6}(L^{-1}) \cdot E_{5,0}(L^{-1}) \cdot E_{3,4}(L^{-2}) \cdot E_{1,2}(L^{-3}) \\ & \cdot E_{4,5}(1) \cdot E_{2,3}(1) \cdot E_{0,1}(1), \end{aligned}$$

and the cost of this matrix is $23n + 2\#\texttt{XOR}(L) + 7\#\texttt{XOR}(L^{-1}) + 3\#\texttt{XOR}(L^{2}) + 3\#\texttt{XOR}(L^{-2}) + 3\#\texttt{XOR}(L^{-3})$.

Applying further optimization via Type-2 elementary matrices by Algorithm 3, we obtain the following matrix pattern:

$$\begin{aligned} M = {} & E_{0,2}(1) \cdot E_{5,1}(1) \cdot E_{1,0}(1) \cdot E_{0,6}(L^{-1}) \cdot E_{4,1}(1) \cdot E_{6,4}(L^{-1}) \\ & \cdot E_{2,6}(1) \cdot E_{2}(L) \cdot E_{4,5}(1) \cdot E_{3,0}(L) \cdot E_{5,2}(L^{-1}) \cdot E_{1,3}(1) \\ & \cdot E_{1}(L) \cdot E_{0,4}(L) \cdot E_{4}(L^{-1}) \cdot E_{3,5}(L^{-3}) \cdot E_{4,1}(L^{-3}) \cdot E_{2,0}(L) \\ & \cdot E_{6,3}(L^{-1}) \cdot E_{0,6}(1) \cdot E_{0}(L) \cdot E_{5,0}(L^{-1}) \cdot E_{3,4}(L^{-2}) \cdot E_{1,2}(L^{-3}) \\ & \cdot E_{4,5}(1) \cdot E_{2,3}(1) \cdot E_{0,1}(1). \end{aligned} \tag{9}$$

Now the cost to implement M is about $23n + 6\#\texttt{XOR}(L) + 6\#\texttt{XOR}(L^{-1}) + \#\texttt{XOR}(L^{-2}) + 3\#\texttt{XOR}(L^{-3})$.

Lastly, we choose an $n \times n$ binary matrix L to check if its corresponding matrix M is MDS by computing the minors of M and checking if they are non-zero according to Theorem Lemma 1. Finally, we obtain 7×7 MDS matrices with word sizes $n = 8, 16, 32$ and 64 bits, which are summarized in Table 8.

Table 8: Construction of 7×7 MDS matrices with word sizes $n = 8, 16, 32$ and 64 bits

n	Field/Ring	Pattern	L	#XOR
8	$\mathbb{F}_{2^8}$/0x187	Exp. (9)	$[[8],[1,2],[2,8],\ldots,[7]]^*$	227
16	$GL(16,\mathbb{F}_2)$	Exp. (9)	$[[2,16],[1],[2,5],[3],\ldots,[15]]$	414
32	$GL(32,\mathbb{F}_2)$	Exp. (9)	$[[11,32],[1],[2],\ldots,[31]]$	759
64	$GL(64,\mathbb{F}_2)$	Exp. (9)	$[[15,64],[1],[2],\ldots,[63]]$	1495

* L is the companion matrix of $x^8 + x^7 + x^2 + x + 1 \in \mathbb{F}_2[x]$.

4.5 Construction of 8×8 MDS Matrices

Firstly, according to Algorithm 1, we can construct potential 8×8 MDS matrix patterns by 26 Type-3 elementary matrices. For example, consider an MDS matrix pattern as follows:

$$\begin{aligned} M = {} & E_{5,6}(L^{\lambda_{26}}) \cdot E_{1,2}(L^{\lambda_{25}}) \cdot E_{4,5}(L^{\lambda_{24}}) \cdot E_{2,3}(L^{\lambda_{23}}) \cdot E_{0,1}(L^{\lambda_{22}}) \\ & \cdot E_{6,7}(L^{\lambda_{21}}) \cdot E_{7,4}(L^{\lambda_{20}}) \cdot E_{5,2}(L^{\lambda_{19}}) \cdot E_{3,0}(L^{\lambda_{18}}) \cdot E_{1,6}(L^{\lambda_{17}}) \\ & \cdot E_{2,7}(L^{\lambda_{16}}) \cdot E_{0,5}(L^{\lambda_{15}}) \cdot E_{6,3}(L^{\lambda_{14}}) \cdot E_{4,1}(L^{\lambda_{13}}) \cdot E_{1,2}(L^{\lambda_{12}}) \\ & \cdot E_{7,0}(L^{\lambda_{11}}) \cdot E_{5,6}(L^{\lambda_{10}}) \cdot E_{3,4}(L^{\lambda_{9}}) \cdot E_{6,1}(L^{\lambda_{8}}) \cdot E_{4,7}(L^{\lambda_{7}}) \\ & \cdot E_{2,5}(L^{\lambda_{6}}) \cdot E_{0,3}(L^{\lambda_{5}}) \cdot E_{7,6}(L^{\lambda_{4}}) \cdot E_{5,4}(L\lambda_{3}) \cdot E_{3,2}(L^{\lambda_{2}}) \cdot E_{1,0}(L^{\lambda_{1}}), \end{aligned}$$

where L and exponents λ_i $(i = 1, 2, \ldots, 26)$ are symbolic variables and they are unknown. Now the cost of this pattern is $26n + \sum_{i=1}^{26}$ #XOR(L^{λ_i}).

Next, we minimize the sum of the exponents $\sum_{i=1}^{26} |\lambda_i|$ by automatically searching suitable values for λ_i, $i = 1, 2, \ldots, 26$ based on SAT/SMT according to Algorithm 2 over a fixed finite field. However, we cannot find a satisfying assignment for the symbolic exponents over $\mathbb{F}_{2^4}$. Therefore, we try to find a satisfying assignment over $\mathbb{F}_{2^8}$. As a result, we find that one of the best exponent instantiations over $\mathbb{F}_{2^8}$/0x1c3 is

$$\begin{cases} \lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0 \qquad \lambda_5 = \lambda_6 = \lambda_7 = \lambda_8 = 3 \\ \lambda_{13} = \lambda_{14} = \lambda_{15} = \lambda_{16} = \lambda_{21} = \lambda_{22} = \lambda_{23} = \lambda_{24} = -1 \\ \lambda_9 = \lambda_{10} = \lambda_{11} = \lambda_{12} = 1 \qquad \lambda_{25} = \lambda_{26} = -5 \\ \lambda_{17} = \lambda_{18} = \lambda_{19} = \lambda_{20} = -2 \end{cases}.$$

Now the matrix M is

$$\begin{aligned} M = {} & E_{5,6}(L^{-5}) \cdot E_{1,2}(L^{-5}) \cdot E_{4,5}(L^{-1}) \cdot E_{2,3}(L^{-1}) \cdot E_{0,1}(L^{-1}) \\ & \cdot E_{6,7}(L^{-1}) \cdot E_{7,4}(L^{-2}) \cdot E_{5,2}(L^{-2}) \cdot E_{3,0}(L^{-2}) \cdot E_{1,6}(L^{-2}) \\ & \cdot E_{2,7}(L^{-1}) \cdot E_{0,5}(L^{-1}) \cdot E_{6,3}(L^{-1}) \cdot E_{4,1}(L^{-1}) \cdot E_{1,2}(L) \\ & \cdot E_{7,0}(L) \cdot E_{5,6}(L) \cdot E_{3,4}(L) \cdot E_{6,1}(L^{3}) \cdot E_{4,7}(L^{3}) \\ & \cdot E_{2,5}(L^{3}) \cdot E_{0,3}(L^{3}) \cdot E_{7,6}(1) \cdot E_{5,4}(1) \cdot E_{3,2}(1) \cdot E_{1,0}(1), \end{aligned}$$

and the cost of this matrix is $26n + 4$#XOR$(L) + 8$#XOR$(L^{-1}) + 4$#XOR$(L^{-2}) + 4$#XOR$(L^3) +$ 2#XOR(L^{-5}).

Applying further optimization via Type-2 elementary matrices by Algorithm 3, we obtain the following matrix pattern:

$$\begin{aligned} M = {} & E_{5,6}(L^{-5}) \cdot E_{1,2}(L^{-5}) \cdot E_{4,5}(L^{-1}) \cdot E_{2,3}(1) \cdot E_{0,1}(1) \\ & \cdot E_{6,7}(L^{-1}) \cdot E_{7,4}(L^{-2}) \cdot E_{5,2}(L^{-2}) \cdot E_{3,0}(L^{-4}) \cdot E_{1,6}(L^{-2}) \\ & \cdot E_{2,7}(L^{-1}) \cdot E_{0,5}(1) \cdot E_{6,3}(1) \cdot E_{4,1}(L^{-1}) \cdot E_{1,2}(L) \\ & \cdot E_{7,0}(1) \cdot E_{0}(L) \cdot E_{5,6}(L) \cdot E_{3,4}(1) \cdot E_{3}(L^{-1}) \cdot E_{6,1}(L^{3}) \cdot E_{4,7}(L^{3}) \\ & \cdot E_{2,5}(L^{3}) \cdot E_{0,3}(L^{3}) \cdot E_{7,6}(1) \cdot E_{5,4}(1) \cdot E_{3,2}(1) \cdot E_{1,0}(1). \end{aligned} \tag{10}$$

Now the cost to implement M is about $26n + 3\texttt{\#XOR}(L) + 5\texttt{\#XOR}(L^{-1}) + 3\texttt{\#XOR}(L^{-2}) + \texttt{\#XOR}(L^{-4}) + 4\texttt{\#XOR}(L^{3}) + 2\texttt{\#XOR}(L^{-5})$.

In order to construct the MDS matrix with large-scale but lower area, we have the following exponent instantiation:

$$\begin{cases} \lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = \lambda_6 = \lambda_7 = \lambda_8 = \lambda_{25} = \lambda_{26} = 0 \\ \lambda_9 = \lambda_{10} = \lambda_{11} = \lambda_{12} = \lambda_{17} = \lambda_{18} = \lambda_{19} = \lambda_{20} = 1 \\ \lambda_{13} = \lambda_{14} = \lambda_{15} = \lambda_{16} = 2 \qquad \lambda_{21} = \lambda_{22} = \lambda_{23} = \lambda_{24} = -1 \end{cases},$$

applying further optimization via Type-2 elementary matrices by Algorithm 3, we obtain the following matrix pattern:

$$\begin{aligned} M = {} & E_{5,6}(1) \cdot E_{1,2}(1) \cdot E_{4,5}(1) \cdot E_{2,3}(L^{-1}) \cdot E_{0,1}(1) \\ & \cdot E_{6,7}(L^{-1}) \cdot E_{7,4}(L) \cdot E_{4}(L) \cdot E_{5,2}(L) \cdot E_{3,0}(1) \cdot E_{0}(L) \cdot E_{1,6}(L) \\ & \cdot E_{2,7}(L^2) \cdot E_{0,5}(L^2) \cdot E_{6,3}(L^2) \cdot E_{4,1}(L^2) \cdot E_{1,2}(L) \\ & \cdot E_{7,0}(L) \cdot E_{5,6}(L) \cdot E_{3,4}(L) \cdot E_{6,1}(1) \cdot E_{4,7}(1) \\ & \cdot E_{2,5}(1) \cdot E_{0,3}(1) \cdot E_{7,6}(1) \cdot E_{5,4}(1) \cdot E_{3,2}(1) \cdot E_{1,0}(1). \end{aligned} \tag{11}$$

Now the cost of this matrix is $26n + 8\texttt{\#XOR}(L) + 2\texttt{\#XOR}(L^{-1}) + 4\texttt{\#XOR}(L^2)$.

Finally, we find 8×8 MDS matrices with word sizes $n = 8, 16, 32$ and 64 bits, which are summarized in Table 9.

Table 9: Construction of 8×8 MDS matrices with word sizes $n = 8, 16, 32$ and 64 bits

n	Field/Ring	Pattern	L	`#XOR`
8	$\mathbb{F}_{2^8}$/0x1c3	Exp. (10)	$[[8],[1,3],[1,2,3],[3],\ldots,[7]]^*$	281
16	$GL(16,\mathbb{F}_2)$	Exp. (11)	$[[2,16],[1],[2,5],[3],\ldots,[15]]$	452
32	$GL(32,\mathbb{F}_2)$	Exp. (11)	$[[11,32],[1],[2],\ldots,[31]]$	850
64	$GL(64,\mathbb{F}_2)$	Exp. (11)	$[[15,64],[1],[2],\ldots,[63]]$	1682

* L is the companion matrix of $x^8 + x^7 + x^6 + x + 1 \in \mathbb{F}_2[x]$ and it can be implemented by 2 XORs.

Remark. As our method aims to reduce the number of XOR gates so as to minimize the implementation area, it is important to verify whether these theoretical results consistent with the practical performance. As an example, consider the the 4×4 MDS matrix with an XOR cost of 35; we synthesized it using the TSMC 65 nm standard cell library. Theoretically, its area is $35 \times 2.5 = 87.5$ GE, as each XOR gate costs approximately 2.5 GE in this library. In practice, the post-synthesis result is 84 GE, as the circuit is slightly optimized by the tool itself. This confirms that the theoretical area of matrices is consistent with the practical area to a certain extent.

5 Conclusion

In this paper, we propose a framework to construct MDS matrices based on matrix decomposition technique. Our method combines symbolic pattern generation, SMT-based exponent optimization and greedy strategy to minimize the implementation area cost while preserving the MDS property. Experimental results show that this method achieves competitive performance across various dimensions and word sizes. However, there are still several limitations: firstly, for certain matrix dimensions such as 6×6, 7×7 and 8×8, we were unable to find MDS matrices with word size $n = 4$ with the current search strategy; secondly, the trade-off between SMT-based exponent optimization and Type-2 matrix insertion is nontrivial; thirdly, we limited our search to matrix dimensions up to $m = 8$ because the search space and required computational resources grow rapidly with increasing m. In the future, we hope these limitations will be fixed by some new techniques.

Acknowledgments

This resarch is supported by the Open Project Program from Key Laboratory of Cryptologic Technology and Information Security (Ministry of Education), Shandong University and the Fundamental Research Funds for the Provincial Universities of Zhejiang (GK259909299001-009).

References

[BKL16] Christof Beierle, Thorsten Kranz, and Gregor Leander. Lightweight multiplication in GF(2^n) with applications to MDS matrices. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 625–653. Springer, Berlin, Heidelberg, August 2016.

[CL19] Victor Cauchois and Pierre Loidreau. On circulant involutory MDS matrices. *DCC*, 87(2-3):249–260, 2019.

[DL18] Sébastien Duval and Gaëtan Leurent. MDS matrices with lightweight circuits. *IACR Trans. Symm. Cryptol.*, 2018(2):48–78, 2018.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[GPS22] Kishan Chand Gupta, Sumit Kumar Pandey, and Susanta Samanta. Construction of recursive MDS matrices using DLS matrices. In Lejla Batina and Joan Daemen, editors, *AFRICACRYPT 22*, volume 2022 of *LNCS*, pages 3–27. Springer, Cham, July 2022.

[GPV17] Kishan Chand Gupta, Sumit Kumar Pandey, and Ayineedi Venkateswarlu. On the direct construction of recursive MDS matrices. *DCC*, 82(1-2):77–94, 2017.

[GR13] Kishan Chand Gupta and Indranil Ghosh Ray. On constructions of involutory MDS matrices. In Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *AFRICACRYPT 13*, volume 7918 of *LNCS*, pages 43–60. Springer, Berlin, Heidelberg, June 2013.

[ISO21] ISO/IEC 18033-3:2010/Amd 1:2021 – Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers – Amendment 1: SM4, 2021. Amendment 1 to ISO/IEC 18033-3:2010.

[JPST17] Jérémy Jean, Thomas Peyrin, Siang Meng Sim, and Jade Tourteaux. Optimizing implementations of lightweight building blocks. *IACR Trans. Symm. Cryptol.*, 2017(4):130–168, 2017.

[KM11] Masanobu Katagi and Shiho Moriai. The 128-bit blockcipher CLEFIA. *RFC*, 6114:1–33, 2011.

[KPPY14] Khoongming Khoo, Thomas Peyrin, Axel York Poschmann, and Huihui Yap. FOAM: Searching for hardware-optimal SPN structures and components with a fair comparison. In Lejla Batina and Matthew Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 433–450. Springer, Berlin, Heidelberg, September 2014.

[KSV19] Abhishek Kesarwani, Santanu Sarkar, and Ayineedi Venkateswarlu. Exhaustive search for various types of MDS matrices. *IACR Trans. Symm. Cryptol.*, 2019(3):231–256, 2019.

[LSS$^+$19] Shun Li, Siwei Sun, Danping Shi, Chaoyun Li, and Lei Hu. Lightweight iterative MDS matrices: How small can we go? *IACR Trans. Symm. Cryptol.*, 2019(4):147–170, 2019.

[LW16] Yongqiang Li and Mingsheng Wang. On the construction of lightweight circulant involutory MDS matrices. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 121–139. Springer, Berlin, Heidelberg, March 2016.

[LXZZ21] Da Lin, Zejun Xiang, Xiangyong Zeng, and Shasha Zhang. A framework to optimize implementations of matrices. In Kenneth G. Paterson, editor, *CT-RSA 2021*, volume 12704 of *LNCS*, pages 609–632. Springer, Cham, May 2021.

[PSA$^+$18] Meltem Kurt PehlIvanoõlu, Muharrem Tolga Sakalli, Sedat Akleylek, Nevcihan Duru, and Vincent Rijmen. Generalisation of hadamard matrix to generate involutory MDS matrices for lightweight cryptography. *IET Inf. Secur.*, 12(4):348–355, 2018.

[SDMO12] Mahdi Sajadieh, Mohammad Dakhilalian, Hamid Mala, and Behnaz Omoomi. On construction of involutory MDS matrices from vandermonde matrices in $GF(2^q)$. *DCC*, 64(3):287–308, 2012.

[SDMS12] Mahdi Sajadieh, Mohammad Dakhilalian, Hamid Mala, and Pouyan Sepehrdad. Recursive diffusion layers for block ciphers and hash functions. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 385–401. Springer, Berlin, Heidelberg, March 2012.

[SM21] Mahdi Sajadieh and Mohsen Mousavi. Construction of MDS matrices from generalized Feistel structures. *DCC*, 89(7):1433–1452, 2021.

[SS16] Sumanta Sarkar and Habeeb Syed. Lightweight diffusion layer: Importance of Toeplitz matrices. *IACR Trans. Symm. Cryptol.*, 2016(1):95–113, 2016.

[SS17] Sumanta Sarkar and Habeeb Syed. Analysis of toeplitz MDS matrices. In Josef Pieprzyk and Suriadi Suriadi, editors, *ACISP 17, Part II*, volume 10343 of *LNCS*, pages 3–18. Springer, Cham, July 2017.

[TTKS18] Dylan Toh, Jacob Teo, Khoongming Khoo, and Siang Meng Sim. Lightweight MDS serial-type matrices with minimal fixed XOR count. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 18*, volume 10831 of *LNCS*, pages 51–71. Springer, Cham, May 2018.

[VKS22] Ayineedi Venkateswarlu, Abhishek Kesarwani, and Sumanta Sarkar. On the lower bound of cost of MDS matrices. *IACR Trans. Symm. Cryptol.*, 2022(4):266–290, 2022.

[WLTZ23] Shi Wang, Yongqiang Li, Shizhu Tian, and Xiangyong Zeng. Four by four MDS matrices with the fewest XOR gates based on words. *Adv. Math. Commun.*, 17(4):845–872, 2023.

[WWW13] Shengbao Wu, Mingsheng Wang, and Wenling Wu. Recursive diffusion layers for (lightweight) block ciphers and hash functions. In Lars R. Knudsen and Huapeng Wu, editors, *SAC 2012*, volume 7707 of *LNCS*, pages 355–371. Springer, Berlin, Heidelberg, August 2013.

[XZL+20] Zejun Xiang, Xiangyong Zeng, Da Lin, Zhenzhen Bao, and Shasha Zhang. Optimizing implementations of linear layers. *IACR Trans. Symm. Cryptol.*, 2020(2):120–145, 2020.

[YWS+24] Yufei Yuan, Wenling Wu, Tairong Shi, Lei Zhang, and Yu Zhang. A framework to improve the implementations of linear layers. *IACR Trans. Symmetric Cryptol.*, 2024(2):322–347, 2024.

A Appendix

Three types of elementary matrices are represented as follows:

$$E_{i,j} = \left[\begin{array}{ccccccc} 1 &&&&&& \\ &\ddots&&&&& \\ &&0&\cdots&1&& \\ &&\vdots&\ddots&\vdots&& \\ &&1&\cdots&0&& \\ &&&&&\ddots& \\ &&&&&&1 \end{array}\right]\begin{array}{c} \\ \\ j \\ \\ i \\ \\ \\ \end{array} \qquad E_i(\alpha) = \left[\begin{array}{ccccccc} 1 &&&&&& \\ &\ddots&&&&& \\ &&1&&&& \\ &&&\alpha&&& \\ &&&&1&& \\ &&&&&\ddots& \\ &&&&&&1 \end{array}\right]\begin{array}{c} \\ \\ \\ i \\ \\ \\ \\ \end{array}$$

$$E_{i,j}(\beta) = \left[\begin{array}{ccccccc} 1 &&&&&& \\ &\ddots&&&&& \\ &&1&\cdots&0&& \\ &&\vdots&\ddots&\vdots&& \\ &&\beta&\cdots&1&& \\ &&&&&\ddots& \\ &&&&&&1 \end{array}\right]\begin{array}{c} \\ \\ j \\ \\ i \\ \\ \\ \end{array}$$

where i and j are the row indices of the elementary matrix, α and β are non-zero elements in the finite field.