Lightweight MDS Serial-type Matrices with Minimal Fixed XOR Count
(Full version)

Dylan Toh[1], Jacob Teo[1], Khoongming Khoo[2], and Siang Meng Sim[2,3,⋆]

[1] NUS High School of Math and Science, Singapore
[2] DSO National Laboratories, Singapore
kkhoongm@dso.org.sg
[3] Nanyang Technological University, Singapore
ssim011@e.ntu.edu.sg

Abstract. Many block ciphers and hash functions require the diffusion property of Maximum Distance Separable (MDS) matrices. Serial matrices with the MDS property obtain a trade-off between area requirement and clock cycle performance to meet the needs of lightweight cryptography. In this paper, we propose a new class of serial-type matrices called Diagonal-Serial Invertible (DSI) matrices with the sparse property. These matrices have a fixed XOR count (contributed by the connecting XORs) which is half that of existing matrices. We prove that for matrices of order 4, our construction gives the matrix with the lowest possible fixed XOR cost. We also introduce the Reversible Implementation (RI) property, which allows the inverse matrix to be implemented using the similar hardware resource as the forward matrix, even when the two matrices have different finite field entries. This allows us to search for serial-type matrices which are lightweight in both directions by just focusing on the forward direction. We obtain MDS matrices which outperform existing lightweight (involutory) matrices.

Keywords: MDS matrix, Serial matrix, lightweight cryptography, XOR count

1 Introduction

Diffusion[1] is a key property of a secure cipher which refers to the propagation of changes in the input to the entire output. In many ciphers, the diffusion property is brought about by a linear diffusion matrix as part of a round function component. Effectively, a modification of even a single bit in the input results in drastic changes in the output, providing stronger defence against differential and linear cryptanalysis attacks. Hence, matrices known as Maximum Distance Separable (MDS) matrices are commonly used in ciphers [2–5] to maximise the

⋆ Supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

diffusion ability of the diffusion layer. They have the property that the diffusion provided is optimal. Many works, such as [6–9], give elegant ways of constructing MDS matrices recursively.

However, the guarantee of strong diffusion power often results in a high hardware computation cost. Thus, it is necessary to find lightweight MDS matrices that can be incorporated into ciphers while minimising hardware requirements and maximising efficiency. However, due to the size of the search space it is impossible to perform a naive exhaustive search. Thus, various constructions [10–13] have been studied in order to narrow the search space to obtain lightweight MDS matrices. Other methods have also been proposed to increase overall efficiency.

One example is so-called *serial matrices*[5], which utilise a trade-off to reduce hardware requirement while incurring additional time cost. These matrices have the property that their k-th power is MDS (k-MDS), thus by applying the matrix k times in a series of k clock cycles, diffusion ability can still be maximised. [11] proposed the idea of cyclic matrices (generalisation of circulant matrices) in a serial-based implementation to simulate serial matrix implementation and to achieve low hardware cost. In both cases, the trade-off area (XOR count) with throughput (number of clock cycles) is kind of balance and proportional, reducing the XOR count by a factor of k while increasing the clock cycle by a factor of k.

Very often, the search for lightweight diffusion matrix focused on the forward direction and paid little attention to the implementation cost of the inverse matrix (backward direction). Although there are scenarios like OFB, CFB and counter mode which only requires the block cipher encryption to be lightweight, in other cases where both encryption and decryption are required, such matrices tends to pay more for its inverse matrix. There are other works which attempt to overcome this problem by considering involution (self-inverse) matrices like in [10, 11, 13]. In this case, both the forward and backward direction will cost the same. However, it often comes with a higher cost because of the involution restriction.

In this work, we aim to search for new serial-type matrices which outperform existing lightweight matrices and also have efficient implementation in both forward and backward direction. This will be useful for constrained devices where lightweight implementation is required but high throughput is not necessary.

Contributions. We propose a new class of serial-type matrices known as *Diagonal-Serial Invertible* (DSI) matrices. This matrix is potentially k-MDS while maintaining a low original weight, which can increase the probability of finding lightweight matrices possessing the targeted properties. By introducing the *sparse* condition for DSI matrices, we actually made a favourable trade-off between the area and clock cycle; we reduced the XOR count by almost a factor of $2k$ at a cost of k clock cycles.

We introduce the concept of Reversible Implementation (RI) property which allows us to better understand the implementation of serial-type matrices. We show that its implementation cost for the backward direction can be as low as the

forward direction, giving us the advantage of having efficient implementation for both forward and backward directions. Because of this RI property, we can focus our search on lightweight matrices in the forward direction without worrying its inverse cost or to limit our search to involution matrices.

Our construction led us to finding new lightweight serial-type matrices, which are lighter than existing diffusion matrices in serialised implementation.

Lastly, we prove for diffusion matrices of order 4 that our sparse DSI matrices achieve the lowest possible fixed cost implementation. Meaning that there does not exist other serial-type matrix construction that would be lighter than our sparse DSI construction.

Organisation. We give the preliminaries in Section 2, introduce our new serial-type matrix construction and its properties in Section 3. Next, we introduce the concept of the RI property and describe how the implementation cost of a diffusion matrix is evaluated in Section 4, and present our search results in Section 5. Finally, we prove optimality for our sparse DSI matrix in Section 6 and end with our conclusion and some thoughts about future work in Section 7.

2 Preliminaries

In this section, we give a preliminary overview of concepts and definitions used in the rest of the paper.

2.1 Finite fields and MDS matrices

We denote by $\mathrm{GF}(2^n)$ the finite field with 2^n elements. It is isomorphic to polynomials in $\mathrm{GF}(2)[X]$ modulo an irreducible polynomial $p(X)$ of degree n. The elements of $\mathrm{GF}(2^n)$ may be written in two ways: in polynomial representation, $\sum b_i X^i$ or in bitwise representation, $b_{n-1}b_{n-2}...b_2b_1b_0$, where $b_i \in \mathrm{GF}(2)$. For example, in $\mathrm{GF}(2^8)$, the 8-bit string 11100001 corresponds to the polynomial $X^7 + X^6 + X^5 + 1$, written `0xe1` in hexadecimal.

The addition operation on $\mathrm{GF}(2^n)$ is simply the bitwise XOR on the coefficients of the polynomial representation of the elements. The multiplication of two elements is the modulo $p(X)$ reduction of the product of the polynomial representations of the two elements. For simplicity, we append the irreducible polynomial in hexadecimal form to the finite field. For instance, suppose $p(X) = X^4 + X^1 + 1$ is the modulo reduction of the product of field elements in $\mathrm{GF}(2^4)$, we denote the finite field as $\mathrm{GF}(2^4)/\mathtt{0x13}$.

Definition 1. *[11] The* ***branch number*** *of a matrix M of order k over finite field $\mathrm{GF}(2^n)$ is the minimum number of nonzero components of the input vector v and output vector $u = M \cdot v$ as we range over all nonzero $v \in [\mathrm{GF}(2^n)]^k$.*

Using matrices with high branch number for the diffusion layer of block ciphers protect them against differential and linear cryptanalysis (protecting against the latter requires the transpose of the diffusion matrix instead to have a high branch number).

Definition 2. *[14] A **maximum distance separable (MDS)** matrix of order k is a matrix that attains the optimal branch number $k+1$.*

Fixing the input vector to have only 1 nonzero element, the output vector will have at best, all its k entries nonzero; therefore the branch number is bounded above by $k+1$.

Definition 3. *A matrix of order k is **q-MDS** if it is MDS when raised to the q-th power.*

Such matrix is also known as **recursive MDS** matrix, but since we will be discussing cases where $q \neq k$ (see Section 6), we chose the notation q-MDS for clearer indication of the number of iterations.

The following proposition is used to check if a matrix satisfies the MDS property:

Proposition 1. *[15] A matrix is MDS if and only if its square submatrices are all nonsingular.*

Corollary 1. *[2] A matrix is MDS if and only if its transpose is also MDS.*

This means that an MDS matrix can best defend against both differential and linear cryptanalysis. We thus restrict our search to MDS matrices.

Proposition 2. *[11] For any permutation matrices P and Q, the branch numbers of these two matrices M and PMQ are the same.*

This provides some symmetry in terms of general construction in the later parts of the paper.

Proposition 3. *The branch numbers of M and M^{-1} are the same for any invertible matrix M.*

Proof. There is a one-to-one correspondence between (input vector, output vector) ordered pairs (u, v) for multiplication with M (where $v = Mu$) and (input vector, output vector) ordered pairs (v, u) for multiplication with M^{-1} (where $v = M^{-1}u$); therefore the branch numbers are consequently identical.

2.2 XOR count

The way to perceive and estimate the implementation cost of the diffusion layer has evolved over time. It was a common belief that finite field elements with low Hamming weight has lower hardware implementation cost. In 2014, the authors of [16] proposed estimating the implementation cost by counting the number of XOR gates (denoted as d-XOR [17]) needed to implement the field element from its multiplication matrix. They also showed that, unlike the common belief, higher Hamming weight elements may also have low implementation cost. Several work [11, 10, 13, 18] adopted this metric to estimate the implementation cost of diffusion matrix. An improved metric s-XOR [17], proposed by authors of [17] was introduced to better gauge the implementation cost in practice. In this paper, we adopt this new metric to calculate the implementation cost of the diffusion layer.

Definition 4. *[17] The **s-XOR count** of an element α in $GF(2^n)/p(X)$ (where $p(X)$ is the generator polynomial), is the minimum number of XOR operations for implementing the field element multiplication, where the minimum is taken over all implementation sequences.*

Example 1. [17] Given the finite field $\mathrm{GF}(2^3)/\mathtt{0xb}$, the multiplication of $\alpha = 7$ seen as $(1,1,1) \in [\mathrm{GF}(2)]^3$ can be computed by:

$$\begin{aligned}(1,1,1)(b_2,b_1,b_0) &= (b_2 \oplus b_0, b_2 \oplus b_1, b_1) \oplus (b_1, b_2 \oplus b_0, b_2) \oplus (b_2, b_1, b_0) \\ &= (b_1 \oplus b_0, b_0, b_2 \oplus b_1 \oplus b_0),\end{aligned}$$

where (b_2, b_1, b_0) is an arbitrary element of $\mathrm{GF}(2^3) \cong (\mathrm{GF}(2))^3$. Expressing the same computation as a matrix multiplication, it rewrites as

$$\begin{bmatrix}0 & 1 & 1\\ 0 & 0 & 1\\ 1 & 1 & 1\end{bmatrix} \begin{bmatrix}b_2\\ b_1\\ b_0\end{bmatrix} = \begin{bmatrix}b_1 \oplus b_0\\ b_0\\ b_2 \oplus b_1 \oplus b_0\end{bmatrix}.$$

From the multiplication matrix, we can see that d-XOR(α) = 3 XOR count. In practice, one can upward-rotate the input vector components, XOR the second component to the first, followed by XORing the first component to the third to obtain the same desired output. Therefore, we get s-XOR(α) = 2 < d-XOR(α).

In [17], the authors denoted it as s-XOR(α) to distinguish it from the metric proposed in [16] which was denoted as d-XOR(α). Since we are adopting this new metric (s-XOR), we simply use XOR(α) to be concise in this paper. In this paper, we focus on the finite fields $\mathrm{GF}(2^4)/\mathtt{0x13}$ and $\mathrm{GF}(2^8)/\mathtt{0x1c3}$, where most of the lightweight diffusion matrices are found. In Appendix B, we present the XOR counts of these field elements obtained using methods described in [17].

2.3 Some MDS matrix constructions

There have been various constructions to search for lightweight MDS matrices of different sizes, where the inverse is also lightweight for some of them. The authors of [10] considered lightweight Hadamard matrices which have the desirable property of being involutory (if the row sum is 1). This removes the ambiguity of the elements of the inverse matrix, and in some implementation designs, save on implementation cost (as the same circuit for matrix multiplication used in encryption is reused in decryption). Hadamard matrices are only defined on matrices with order a power of 2 where lightweight involutory and non-involutory Hadamard matrices of order 4 and 8 have been found.

Cyclic matrices where each row is a same cyclic permutation of the row above is used to construct lightweight matrices in [11]. In that paper, the authors considered the special case of left-circulant matrices where the permutation is a left cyclic shift. Furthermore, it was proven that involutory left-circulant matrices do not exist if the order is a power of 2. In [12], the authors constructed lightweight left-circulant matrices by optimizing the implementation of a single

finite field element multiplication, where they used the s-XOR metric (described in Section 2.2) and optimal choice of finite field basis to reduce overall XOR count.

In some applications, cyclic matrices can be implemented in a serialized way to trade-off throughput (number of clock cycles) with area (XOR count). This is because the entries in each row is rearrangement of the previous row under the same permutation. It is still an open problem if Hadamard matrices can be serialized efficiently like cyclic matrices.

Such serialization implementation is inspired by serial matrices which were first proposed in [5] as part of the PHOTON hash function. These matrices of order k are k-MDS. This allows the designer to trade-off between lower hardware area requirement and more clock cycles, where the serial matrix is reused k times, once per clock cycle. In this paper, we shall explore this concept further and find ways to construct lighter serial-type matrices.

3 Diagonal-Serial Invertible Matrices

First, let us recall the matrix structure of serial matrices. To distinguish it from the other serial-type matrices that we studied in this paper, we shall call this matrix a Linear Feedback Serial (LFS) matrix.

The LFS matrix $L = LFS(z_0, z_1, ..., z_{k-1})$ from [5] is of specific interest in this study. It's expression is shown below:

$$L_{ij} = \begin{cases} z_j, & i = k-1 \\ 1, & i+1 = j \\ 0, & \text{otherwise.} \end{cases}$$

Because $LFS(z_0, z_1, ..., z_{k-1})$ corresponds to a Linear Feedback Shift Register (LFSR) where the feedback taps are given by the last row of the matrix, hence the name.

Properties of the LFS matrices have been investigated in [19], among which is the lightweight expression of the inverse, L^{-1}. In particular, the inverse matrix of a LFS matrix has the following form:

$$L^{-1}_{ij} = \begin{cases} \frac{z_{j+1}}{z_0}, & i = 0, z_k = 1 \\ 1, & i = j+1 \\ 0, & \text{otherwise.} \end{cases}$$

From this expression, the authors of [19] concluded that if $z_0 = 1$, then both a LFS matrix and its inverse has the same finite field entries $z_1, z_2, \ldots, z_{k-1}$ which will lead to both matrices requiring the same hardware resource to implement. We show later in this paper a new technique that allows us to implement LFS matrix and its inverse with the same hardware resources even when $z_0 \neq 1$.

3.1 Diagonal-Serial Invertible (DSI) matrix

Definition 5. *A **Diagonal-Serial Invertible (DSI)** matrix* $D = (D_{ij})_{1 \le i,j \le k} \in [GF(2^n)]^{k \times k}$*, is determined by 2 vectors,* $\mathbf{a} = (a_i)_{1 \le i \le k} \in [GF(2^n) \backslash \{0\}]^k$ *and* $\mathbf{b} = (b_i)_{1 \le i \le k-1} \in [GF(2^n)]^{k-1}$*, as follows*[4]*:*

$$D_{ij} = \begin{cases} a_1, & i = 1, j = k \\ a_i, & i = j + 1 \\ b_i, & i = j \le k - 1 \\ 0, & \text{otherwise.} \end{cases}$$

Example 2. The general DSI matrix $D = DSI(\mathbf{a}, \mathbf{b})$ for $k = 6$ is the following:

$$D = \begin{pmatrix} b_1 & 0 & 0 & 0 & 0 & a_1 \\ a_2 & b_2 & 0 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & 0 & 0 & 0 \\ 0 & 0 & a_4 & b_4 & 0 & 0 \\ 0 & 0 & 0 & a_5 & b_5 & 0 \\ 0 & 0 & 0 & 0 & a_6 & 0 \end{pmatrix}$$

The design is motivated by the $LFS(z_0, z_1, ..., z_{k-1})$ matrix construction. Keeping to the structure of a permutation matrix, the underlying intuition is that pairwise linear combinations of rows will provide a higher diffusion power. We will first prove some elementary properties of the DSI matrix:

Theorem 1. *Every DSI matrix* $D = DSI(\mathbf{a}, \mathbf{b})$ *is invertible.*

Proof. We have $det(D) = D_{1k} \cdot det(M_{1k}) = a_1 \cdot det(M_{1k})$ by cofactor expansion along the rightmost column (where M_{1k} is the matrix formed by removing the topmost row and rightmost column); M_{1k} is upper triangular thus its determinant is simply the product of its diagonal entries, $det(M_{1k}) = a_2 a_3 a_4 \dots a_k$. Therefore $det(D) = a_1 a_2 a_3 \dots a_k \neq 0$ $(\forall i, a_i \neq 0)$ and D is invertible.

To express the q-th power of the general DSI matrix $D = DSI(\mathbf{a}, \mathbf{b})$, we view it as a weighted adjacency matrix to a directed graph with vertices labeled 1 to k (D_{ij} is the weight of the directed edge from vertex i to vertex j). Then for any $q \in \mathbb{N}$, we have:

$$(D^q)_{ij} = \sum_{\text{length } q \text{ paths from } i \text{ to } j} \text{(product of all weights along the path)}$$

with the sum taken over all all paths of length q from vertex i to vertex j. In the above expression, we note that an edge with weight 0 will never contribute to the sum; therefore edges with weight 0 may be added or removed without consequence.

[4] The indices starts from 1 for the convenience of latter discussions.

For the ease of notation, we denote $b_k = 0$, and $P_i(\{s_1, s_2, \ldots, s_{|S|}\}) = \sum_{p_1+p_2+\cdots+p_{|S|}=i;\ p_j \geq 0} s_1^{p_1} s_2^{p_2} \ldots s_{|S|}^{p_{|S|}}$; then we have, for a general DSI matrix D:

$$(D^k)_{ij} = \begin{cases} P_{i-j}(\{b_t | t \in \{j, \ldots, i\}) \cdot \prod_{u \in \{j-1, \ldots, i\}} a_u, & i > j \\ b_i^k + \prod_{u=1}^{k} a_u, & i = j \\ P_{j-i}(\{b_t | t \in \{1, \ldots, i\} \cup \{j, \ldots, k\}) \cdot \prod_{u \in \{1, \ldots, i\} \cup \{j+1, \ldots, k\}} a_u, & i < j \end{cases} \tag{1}$$

The expressions above are obtained by drawing the associated graph, with vertices 1 to k arranged anticlockwise in a circle; the edge weighted a_i points clockwise from vertex i to vertex $i-1$ (mod k), while the edge weighted b_i points from vertex i to itself. For $i \neq j$, a path of length k from vertex i to vertex j must take the path through the $j - i \pmod k$ clockwise edges $i \to i-1 \to \cdots \to j$, while passing through any $i - j \pmod k$ self-pointing edges along the way. A path of length k from vertex i to itself, on the other hand, may be the one passing through all clockwise edges $i \to i-1 \to \cdots \to i$, or the path obtained by traversing the self-pointing edge $i \to i$ for k times.

The graph also illustrates the symmetry of the matrix; the vertices may be relabeled 1 through k still in an anticlockwise fashion but starting at an arbitrary vertex, which will shift the $b_k = D_{kk} = 0$ element to another element along the main diagonal of the matrix. Note the similarity with Proposition 2, taking $Q^{-1} = P =$ (the permutation matrix corresponding to the permutation of vertices as described).

Example 3. The DSI matrix $D = DSI(\mathbf{a}, \mathbf{b})$ of order 4 is expressed below together with its associated graph, shown in Figure 1, of which it is the weighted adjacency matrix:

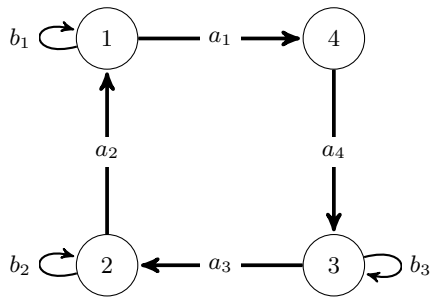
$$D = \begin{pmatrix} b_1 & 0 & 0 & a_1 \\ a_2 & b_2 & 0 & 0 \\ 0 & a_3 & b_3 & 0 \\ 0 & 0 & a_4 & 0 \end{pmatrix}$$

Note that there is no self-pointing edge on vertex 4 because $b_4 = 0$. The fourth power can thus be expressed as shown below:

$$D^4 = \begin{pmatrix} b_1^4 + a_1a_4a_3a_2 & P_1(\{b_1, b_3, b_2\})a_1a_4a_3 & P_2(\{b_1, b_3\})a_1a_4 & P_3(\{b_1\})a_1 \\ P_3(\{b_2, b_1\})a_2 & b_2^4 + a_2a_1a_4a_3 & P_1(\{b_2, b_1, b_3\})a_2a_1a_4 & P_2(\{b_2, b_1\})a_2a_1 \\ P_2(\{b_3, b_2, b_1\})a_3a_2 & P_3(\{b_3, b_2\})a_3 & b_3^4 + a_3a_2a_1a_4 & P_1(\{b_3, b_2, b_1\})a_3a_2a_1 \\ P_1(\{b_3, b_2, b_1\})a_4a_3a_2 & P_2(\{b_3, b_2\})a_4a_3 & P_3(\{b_3\})a_4 & a_4a_3a_2a_1 \end{pmatrix}.$$

We also reach the following result:

Fig. 1: Weighted adjacency graph associated with DSI matrix of order 4



Theorem 2. *Given DSI matrix D of order k, k is the minimum power of D for all entries to have a nonzero algebraic expression (and thus possibly MDS).*

Proof. The shortest path from vertex k to itself is the clockwise path passing through all vertices, $k \to k-1 \to k-2 \to \cdots \to 1 \to k$, of length k, therefore $(D^i)_{kk} = 0$ for all $i < k$. The algebraic expression of all coefficients of D^k have been evaluated above in (1).

3.2 Sparse DSI matrix

However, for the DSI construction, pairwise linear combination of rows would generate maximum diffusion power with inherent redundancy! We can possibly further reduce the number of nonzero entries and consequently lower the fixed cost, as proposed in the following subclass of DSI matrices:

Definition 6. *A DSI matrix $D = DSI(\mathbf{a}, \mathbf{b})$ of order k is **sparse** if $\mathbf{b}$ satisfies:*

$$\begin{cases} b_2 = b_4 = b_6 = \cdots = b_{k-2} = 0, & \text{if } k \text{ is even} \\ b_2 = b_4 = b_6 = \cdots = b_{k-3} = 0, & \text{if } k \text{ is odd} \end{cases}$$

Example 4. An example of sparse DSI matrix of order 4 and 5, denoted as D_4 and D_5 respectively.

$$D_4 = \begin{pmatrix} b_1 & 0 & 0 & a_1 \\ a_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & 0 \\ 0 & 0 & a_4 & 0 \end{pmatrix}, \qquad D_5 = \begin{pmatrix} b_1 & 0 & 0 & 0 & a_1 \\ a_2 & 0 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & 0 & 0 \\ 0 & 0 & a_4 & b_4 & 0 \\ 0 & 0 & 0 & a_5 & 0 \end{pmatrix}.$$

Although sparse DSI matrices do have lesser nonzero entries, a natural question to ask is whether sparse DSI matrices of order k can potentially be k-MDS. Extending the result from Theorem 2, we have the following corollary.

Corollary 2. *Sparse DSI matrices can potentially be k-MDS.*

Proof. By Definition 5, all the a_i's are nonzero. For each appearance of $P_i(S)$ in (1), we want to ensure that there exists nonzero term in the set S. For the case $i > j$, S contains at least two consecutive elements b_j, b_{j+1} , of which at least one term is not set to 0. For the case $i < j$, S always contains b_1 which is nonzero. Thus each appearance of $P_i(S)$ is a nonzero algebraic expression; therefore the algebraic expressions of all entries of D^k are still nonzero for a sparse DSI matrix D.

In fact, if DSI matrix D has 2 consecutive elements b_j, b_{j+1} (with consecutive indices modulo k) with both equal to 0, it follows from (1) that $(D^k)_{(j+1)j} = P_{k-1}(\{b_{j+1}, b_j\})a_{j+1} = 0$ and D^k cannot possibly be MDS; therefore the sparse restriction to the general DSI matrix sets the most number of entries in $\mathbf{b}$ to 0 while still allowing the possibility of D^k being MDS.

In [20], the authors proposed a new serial-type construction based on Type-II Generalized Feistel Structure (GFS). Although this matrix type is similar to our sparse DSI matrix, it is fundamentally different from ours, detailed in Section 6.

4 RI Property and Serial-type Matrices

In this section, we introduce a property called the *Reversible Implementation* (RI) which allows us to understand more about the implementation of serial-type matrices and their inverse. Next, we describe how the implementation cost of the serial-type matrices and their inverse are computed.

4.1 Reversible Implementation (RI) Property

Definition 7. *Given some set of objects S, a function $f : S^n \to S^n$ has the **Reversible Implementation (RI) property** if there exists a sequence of transformations whereby each transformation is either a permutation of the n-tuple components, involution (self-inverse), or the transformation itself has the RI property. Such sequence of transformations is also called the RI sequence.*

Proposition 4. *If a function f has a RI sequence, then there exists an implementation of the inverse function f^{-1} that has the same implementation cost as the RI sequence.*

Proof. Given a RI sequence, we construct a sequence of transformations to implement the inverse function with the same implementation cost as the RI sequence. First, we reverse the entire sequence of transformations, that is starting from the last transformation of the RI sequence. If a transformation is some permutation of the n-tuple components, we implement the inverse permutation on the components. Since permutation is simply rewiring of the circuit in hardware, it is basically free. If a transformation is involution, we apply the exact same transformation with the same implementation cost. If a transformation has the RI property, it has some RI sequence of its own and we can recursively implement its inverse with the same implementation cost.

Example 5. The field multiplication of $7 \in \mathrm{GF}(2^3)/\mathtt{0xb}$ has the RI property as we can see from Example 1, the sequence of transformation:

1. upward-rotate the input vector components,
2. XOR the second component to the first component,
3. XOR the first component to the third component,

is a RI sequence since it consists of permutation and XOR instructions that are involutions. The inverse of this field multiplication can be implemented as follows:

1. XOR the first component to the third component,
2. XOR the second component to the first component,
3. downward-rotate the input vector components,

and it has the same implementation cost of 2 XORs as element 7.

In fact, under the s-XOR metric introduced in [17], we can conveniently conclude that *any nonzero finite field element has the RI property and its inverse has the same XOR count as itself* [5].

It is to note that this RI property is different from a function being involution. While being involution means the same circuit could be reused for forward and backward implementation at a cost of some multiplexers, more multiplexers might be needed to reuse the RI sequence circuit in the reverse order. Instead of reusing of circuit, our RI property is useful for identifying the implementation of the inverse requires the similar hardware resources.

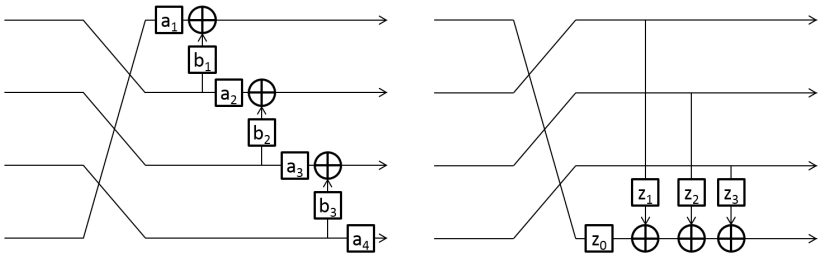
4.2 RI Property in Serial-type Matrices

The sparse matrix structure of serial-type matrices allows us to analyse the sequence of transformations easily. We illustrate the implementation sequence of the two serial-type matrices of order 4 in Figure 2, which can also be generalised to any order k. One can observe that both LFS and DSI matrices have the RI property.

Previous, it was believed that LFS matrix has the same implementation cost for both forward and backward implementation only when $z_0 = 1$ [19]. However, under the s-XOR metric, field element multiplications have the RI property, the implementation cost of z_0 is actually the same as its inverse z_0^{-1}. Therefore, *LFS matrix has the same implementation cost for both forward and backward direction for any nonzero* z_0.

Similar argument holds for DSI matrix. One can implement the inverse of DSI matrix by first updating the last component with field multiplication a_k^{-1}, which as the same implementation cost as a_k. Next, multiply the last component with b_{k-1} and XOR it to the second last component. Repeat these process until the first component. Update the first component with a_1^{-1} and finally upward-rotate the components to obtain the final output vector. The entire process has the same implementation cost as the forward DSI matrix implementation.

[5] This observation has also been pointed out in [12].



(a) Circuit of DSI matrix. (b) Circuit of LFS matrix.

Fig. 2: Circuit of LFS and DSI matrices of order 4.

4.3 Evaluating the implementation cost of Serial-type Matrices

For round-based MDS matrices, a direct consequence of Proposition 1 is that all entries have to be nonzero. Thus, the conventional way of estimating the implementation cost of an MDS matrix over $\mathrm{GF}(2^n)$ is to take the sum of the implementation cost of all nonzero field multiplications (so-called the *variable cost*), and add $(k-1)$ many n-bit XOR count for each row to generate the output components (so-called the *fixed cost*).

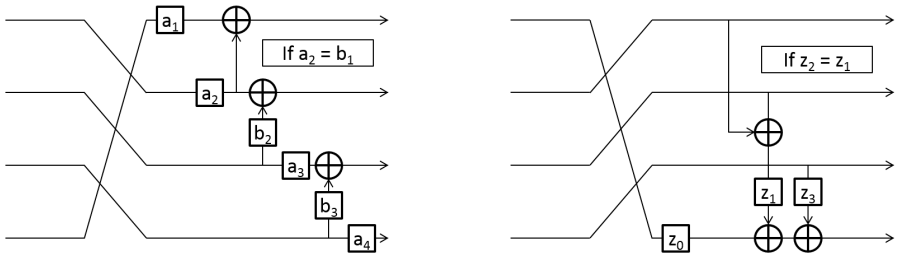
Things are a bit different for serial-type q-MDS matrices, there can be zero entries in the matrix, thus the fixed cost for each row is 1 less than the number of nonzero entries many n-bit XOR count. For the variable cost, we can apply a small technique to save some implementation cost of the field multiplication for some special cases.

Take DSI matrix of order 4 as an example, notice that if $b_1 = a_2$, we can rearrange the order of the field multiplication and the XOR operation so that we only need to compute one field multiplication and not both b_1 and a_2 separately, as shown in Figure 3a. Thus, we only count the implementation cost of b_i when it is not equal to a_{i+1}.

Similar strategy can be applied to LFS matrices, when there are multiple z_i's of the same field element, we can first XOR these branches together before applying a single field multiplication, then XOR to the last component, Figure 3b illustrates an example of LFS matrix of order 4. Therefore, we only need to count the implementation cost of the distinct z_i's in the last row.

Table 1 shows the formula for computing the implementation cost of the entire matrix for various matrices.

While such trick can be applied to these two serial-type matrices, it is non-trivial to apply it to other types of matrices like Hadamard or circulant matrices. Thus, it remains an open question if similar trick can be applied to other matrices too.



(a) Special case of DSI matrix. (b) Special case of LFS matrix.

Fig. 3: Saving implementation cost for serial-type matrices.

Table 1: Total XOR count of various matrices

matrices of order k (over $GF(2^n)$)	XOR count of the entire matrix
Cyclic	$k \cdot \sum XOR(c_i) + k \cdot (k-1) \cdot n$
Hadamard	$k \cdot \sum XOR(h_i) + k \cdot (k-1) \cdot n$
LFS	$\sum_{z_i \mid \forall j<i, z_i \neq z_j} XOR(z_i) + (k-1) \cdot n$
Sparse DSI	$\sum XOR(a_i) + \sum_{b_i \neq a_{i+1}} XOR(b_i) + \lceil k/2 \rceil \cdot n$

where c_i's (resp. h_i's) are the entries in a row of cyclic (resp. Hadamard) matrix.

5 Main Results

In this section, we present new serial-type k-MDS matrices which have lower XOR count than previously known MDS/k-MDS matrices. To obtain lightweight matrices, we searched through $k \times k$ sparse DSI and LFS matrices over the finite field $GF(2^8)$ defined by the irreducible polynomial `0x1c3` for $4 \leq k \leq 8$. To check if a matrix is k-MDS, we raise the matrix to power of k and recursively check that each submatrix is non-singular. Once a submatrix is singular, we exit prematurely and move on to the next candidate. Else, this candidate would be a k-MDS matrix. Besides searching directly in GF(2^8), we also use the subfield construction [16, Section 7.2] where we search over GF(2^4) defined by the irreducible polynomial `0x13`, and interleave two copies to obtain a diffusion matrix with the same branch number over GF(2^8). The search can be completed relatively fast on a personal laptop from a few minutes when $k = 4$ to a few hours when $k = 7$. The lightweight sparse DSI and LFS k-MDS matrices, denoted by $D_{k,n}$ and $L_{k,n}$ respectively, found are listed in Appendix A.

We reiterate that we are considering scenario where lightweight implementation is required but high throughput is not necessary. That is, we do a tradeoff for lower area at the cost of higher clock cycles. For serial-type matrices, such tradeoff is natural and the implementation cost is the XOR count of the entire matrix. For cyclic matrices, for instance circulant/left-circulant matrices, it can be implemented in a serialized manner by implementing one row of the matrix and reusing the same row [11]. Although it is non-trivial to implement Hadamard

matrices in a serialized manner, we give the benefit of the doubt and assume that it can be done too. Therefore, we compare these round-based matrices by the XOR count for one of its row, see Table 2. Note that we are considering only the implementation cost of the matrix, the cost of multiplexer is out of the scope of our discussion.

Table 2: XOR count of various serialized matrices

matrices of order k (over $GF(2^n)$)	XOR count
Cyclic	$\sum XOR(c_i) + (k-1) \cdot n$
Hadamard	$\sum XOR(h_i) + (k-1) \cdot n$
LFS	$\sum_{z_i \mid \forall j<i, z_i \neq z_j} XOR(z_i) + (k-1) \cdot n$
Sparse DSI	$\sum XOR(a_i) + \sum_{b_i \neq a_{i+1}} XOR(b_i) + \lceil k/2 \rceil \cdot n$

where c_i's (resp. h_i's) are the entries in a row of cyclic (resp. Hadamard) matrix.

For fair comparison, we recalculated the implementation cost of the existing matrices under the same metric (s-XOR) as ours when possible. In Table 3 (resp. Table 4), we compare serialized MDS matrices and k-MDS serial-type matrices of order $4 \leq k \leq 8$ over GF(2^4) (resp. GF(2^8)) in both the forward and backward direction. In particular, we compare the DSI and LFS matrices that we found with circulant [13], left-circulant [11, 12], Hadamard [10] and LFS [5, 16, 21] matrices. One may also consider unrolling DSI/LFS matrices to simulate round-based matrices for comparison with cyclic/Hadamard matrices in a round-based implementation scenario. That is to implement k copies of DSI/LFS matrices in series to achieve the MDS property in one clock cycle. The XOR count of all the matrices would simply be k times of what the tables have shown.

For the entries of the diffusion matrix, some literature considered invertible binary matrices rather than the finite field elements, for those cases, we indicate the entry type as GL(n, GF(2)).

5.1 Comparing Matrices where $n = 4$

We ran our search on both sparse DSI and LFS k-MDS matrices of order $4 \leq k \leq 8$ over GF(2^4)/0x13. The results are summarised in Table 3.

In [21], the authors considered 8-MDS matrix of order 4 to extend their search for lighter diffusion matrices. Although they obtained LFS matrix with lower XOR count than other 4-MDS LFS matrices, in serial-based implementation, the number of clock cycle needed is doubled.

Although we could not find sparse DSI k-MDS matrices of order higher than 4, we would like to point out that this is quite common even for other matrix types due to the small field size. As we can see in Table 3 that there are lesser matrices that we can compare as the matrix size increases.

For circulant matrices from [13] and serial-type matrices from [20], the entries of the matrix are non-singular binary matrices of order n. In [12], the irreducible

Table 3: Comparison of MDS/k-MDS matrices for $n = 4$

k	Matrix Type	Field/Ring	Forward	Backward	Reference
4	Hadamard	GF(2^4)/0x13	17	19	[10]
4	Involutory Hadamard	GF(2^4)/0x13	17	17	[10]
4	Involutory Circulant	GL$(4, GF(2))$	17	17	[13]
4	Left-circulant	GF(2^4)	15	–	[12]
4	Circulant	GL$(4, GF(2))$	15	–	[13]
4	Left-circulant	GF(2^4)/0x13	15	29	[11]
4	**LFS**	GL$(4, GF(2))$	15	-	[20]
4	**LFS**	GF(2^4)/0x13	15	15	[16]
4	8-MDS **LFS**	GF(2^4)/0x13	13	13	[21]
4	**GFS**	GL$(4, GF(2))$	10	-	[20]
4	**Sparse DSI** $D_{4,4}$	GF(2^4)/0x13	10	10	This Paper
5	IMDS left-circulant	GF(2^4)/0x13	27	27	[11]
5	Left-circulant	GF(2^4)	24	–	[12]
5	Left-circulant	GF(2^4)/0x13	20	26	[11]
5	**LFS**	GL$(4, GF(2))$	19	-	[20]
5	**LFS** A_{100}	GF(2^4)/0x13	18	18	[5]
5	**LFS** $L_{5,4}$	GF(2^4)/0x13	18	18	This Paper, A_{100}
6	Left-circulant	GF(2^4)/0x13	30	40	[11]
6	**LFS** A_{144}	GF(2^4)/0x13	28	28	[5]
6	**LFS**	GL$(4, GF(2))$	25	-	[20]
6	**LFS** $L_{6,4}$	GF(2^4)/0x13	25	25	This Paper
7	**LFS** A_{196}	GF(2^4)/0x13	31	31	[5]
7	**LFS**	GL$(4, GF(2))$	30	-	[20]
7	**LFS** $L_{7,4}$	GF(2^4)/0x13	30	30	This Paper
8	Involutory Hadamard	GF(2^4)/0x13	53	53	[10]
8	Hadamard	GF(2^4)/0x13	48	56	[10]
8	**LFS** A_{256}	GF(2^4)/0x13	47	47	[5]
8	**LFS**	GF(2^4)/0x13	41	41	[16]
8	**LFS**	GL$(4, GF(2))$	37	-	[20]
8	**LFS** $L_{8,4}$	GF(2^4)/0x13	36	36	This Paper

polynomial is not defined. Therefore, it is unclear how we can obtain the inverse of those matrices.

Besides obtaining the same LFS matrix as [5] for $k = 5$, we found new lightweight serial-type k-MDS matrices that outperform or match the existing lightweight matrices. Although the improvement margin may seem small, it is to note that we are comparing with the state-of-the-art lightweight diffusion matrices and it is non-trivial to outperform any of them. For matrices over GF(2^4), the room for improvement is small due to the small field size, one can see larger improvements for matrices over GF(2^8) in the next section.

5.2 Comparing Matrices where $n = 8$

In addition to running our search on both sparse DSI and LFS k-MDS matrices of order $4 \leq k \leq 8$ over GF(2^8)/0x1c3, we also considered the subfield con-

struction [16, Section 7.2] where we use 2 copies of the serial-type matrices over $\mathrm{GF}(2^4)/\mathsf{0x13}$, denoted by $[\,\cdot\,]^2$, hence doubling the XOR count[6]. The results are summarised in Table 4.

Table 4: Comparison of MDS/k-MDS matrices for $n = 8$

k	Matrix Type	Field/Ring	Forward	Backward	Reference
4	Hadamard	$[\mathrm{GF}(2^4)/\mathsf{0x13}]^2$	2×17	2×19	[10]
4	Involutory Hadamard	$[\mathrm{GF}(2^4)/\mathsf{0x13}]^2$	2×17	2×17	[10]
4	Involutory Circulant	$\mathrm{GL}(8, GF(2))$	33	33	[13]
4	**LFS**	$\mathrm{GF}(2^8)/\mathsf{0x11d}$	33	33	[16]
4	Left-circulant	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	31	75	[11]
4	Left-circulant	$\mathrm{GF}(2^8)$	30	-	[12]
4	8-MDS **LFS**	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	27	27	[21]
4	Circulant	$\mathrm{GL}(8, GF(2))$	27	-	[13]
4	**LFS**	$\mathrm{GL}(8, GF(2))$	27	-	[20]
4	**Sparse DSI** $D_{4,8}$	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	22	22	This Paper
4	**Sparse DSI** $[D_{4,4}]^2$	$[\mathrm{GF}(2^4)/\mathsf{0x13}]^2$	2×10	2×10	This Paper
4	**GFS**	$\mathrm{GL}(8, GF(2))$	18	-	[20]
5	IMDS left-circulant	$\mathrm{GF}(2^8)/\mathsf{0x165}$	65	65	[11]
5	Left-circulant	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	42	90	[11]
5	Left-circulant	$\mathrm{GF}(2^8)$	40	-	[12]
5	**LFS**	$\mathrm{GL}(8, GF(2))$	35	-	[20]
5	**Sparse DSI** $D_{5,8}$	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	31	31	This Paper
6	IMDS left-circulant	$\mathrm{GF}(2^8)/\mathsf{0x165}$	77	77	[11]
6	**LFS** A_{288}	$\mathrm{GF}(2^8)/\mathsf{0x11b}$	57	57	[5]
6	Left-circulant	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	55	115	[11]
6	Left-circulant	$\mathrm{GF}(2^8)$	54	-	[12]
6	**LFS**	$\mathrm{GL}(8, GF(2))$	45	-	[20]
6	**GFS**	$\mathrm{GF}(2^8)$	≥ 42	-	[20]
6	**Sparse DSI** $D_{6,8}$	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	31	31	This Paper
7	IMDS left-circulant	$\mathrm{GF}(2^8)/\mathsf{0x139}$	107	107	[11]
7	Left-circulant	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	66	120	[11]
7	Left-circulant	$\mathrm{GF}(2^8)$	64	-	[12]
7	**LFS**	$\mathrm{GL}(8, GF(2))$	54	-	[20]
7	**Sparse DSI** $D_{7,8}$	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	54	54	This Paper
8	Involutory Hadamard	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	96	96	[10]
8	Hadamard	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	88	179	[10]
8	Left-circulant	$\mathrm{GF}(2^8)$	82	-	[12]
8	Left-circulant	$\mathrm{GF}(2^8)/\mathsf{0x1c3}$	80	160	[11]
8	**LFS** $[L_{8,4}]^2$	$[\mathrm{GF}(2^4)/\mathsf{0x13}]^2$	2×36	2×36	This Paper
8	**LFS**	$\mathrm{GL}(8, GF(2))$	65	-	[20]

[6] We can multiply the XOR counts of all matrices in Table 3 by 2 to get matrices over $\mathrm{GF}(2^8)$ but we do not include most of them in Table 4 to prevent congestion. But we can easily see that the best (sparse DSI) matrices we get directly from $\mathrm{GF}(2^8)/\mathsf{0x1c3}$ do outperform 2 copies of the best matrices over $\mathrm{GF}(2^4)$ for $5 \leq k \leq 7$.

Similar to the case $n = 4$, it is unclear how the inverse of the matrices from [13, 20, 12] are obtained. In [21], the authors considered 8-MDS matrix of order 4 to have lower XOR count. However, in serial-based implementation, this results in more clock cycles and higher latency compared to other serial-type matrices.

For $5 \leq k \leq 7$, we found sparse DSI matrices that outperform existing lightweight matrices in both forward and backward direction. For $k = 6$, the authors of [20] pointed out that there exists 6-MDS GFS matrix over finite field without giving an actual example, assuming that it can be constructed with some field element with the lowest possible XOR count of 3, we obtain, at best, an estimation of XOR count 42.

When $k = 8$, the search space for sparse DSI and LFS matrices are too large to cover and we have not found a k-MDS matrix yet. Similar problem was faced by the authors of [16] when they search for LFS matrices. Nevertheless, we can construct a competitive candidate matrix from $L_{8,4}$ using subfield construction.

Although we did not outperform the matrices from [20] for the case $k = 4, 8$, it is to note that the choice of matrix entry is different. While it is possible to search for lightweight DSI matrices over invertible binary matrices, it requires very different search strategy and it is beyond the scope of our work.

Notice that for non-involution round-based matrices, its inverse could be significantly larger. However for serial-type matrices, we do not have such problem thanks to the RI property. Thus, when the implementation of the backward direction is required, our matrices are more favourable.

6 Advantages of Sparse DSI Matrices

In this section, we look at the reasons behind why sparse DSI matrices tend to yield better results than other matrix types like Hadamard, cyclic and LFS matrices. In addition, we prove that sparse DSI matrix of order 4 has the lowest possible fixed XOR count. Since diffusion matrix of order 4 is probably the most commonly used matrix size for the diffusion layer, sparse DSI matrix is a great candidate for designing lightweight ciphers.

6.1 Reducing the fixed XOR count

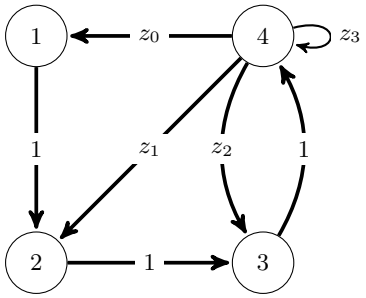
The biggest limitation to MDS matrix was the fixed XOR cost. For matrices like Hadamard or cyclic matrices, a necessary condition for a matrix to be MDS is not to have any zero entries. This means that for each row of the matrix, there are up to k finite field multiplications as variable cost (some may be element 1 which is free), and these k components have to be summed together, incurring a fixed cost of $(k - 1) \cdot n$ XOR count.

While most of the existing work focused on reducing the variable cost by considering various type of matrix structure, LFS matrix was introduced as a trade-off between hardware implementation cost and clock cycle. However, the fixed cost in the last row of the LFS matrix remains the same as we can see from the following:

Theorem 3. *If a LFS matrix is k-MDS, then $z_i \neq 0$ for all i.*

Proof. Let matrix $L = LFS(z_0, z_1, ..., z_{k-1})$ be the weighted adjacency matrix to directed graph G (with vertices correspondingly labeled 1 to k, and L_{ij} being the weight of the directed edge $i \to j$). This graph will have edges $i \to i+1$ of weight 1 for $i = 1, 2, 3, ..., k-1$, as well as edges $k \to i$ of weight z_{i-1} for $i = 1, 2, 3, ..., k$. Figure 4 shows an example of a LFS matrix of order 4.

Fig. 4: Weighted adjacency graph associated with LFS matrix of order 4



Thus for each $a = 1, 2, 3, ..., k$, the only path of length k from vertex 1 to vertex a is the path $1 \to 2 \to 3 \to 4 \to ... \to k \to a$. Thus $(L^k)_{1a} = 1^{k-1} \cdot z_{a-1} = z_{a-1}$.

If L^k is MDS, then all its elements are nonzero; it follows directly that $z_i \neq 0$ for all i.

This means that the fixed cost of a LFS k-MDS matrix is necessarily $(k-1) \cdot n$, similar to the fixed cost of a row of MDS Hadamard or cyclic matrix. However, we managed to overcome this limitation by using sparse DSI matrix. The fundamental reason is that sparse DSI matrix can have lesser connecting XORs than other existing matrix structures while still be potentially k-MDS.

As we can see from Table 2, the fixed XOR count of a sparse DSI matrix (contributed by the connecting XORs independent of the choice of matrix entries) is approximately half that of cyclic, Hadamard and LFS matrices. Although there are more a_i and b_i entries in a DSI matrix than in a row of the other matrices, the overall XOR count is greatly compensated by the fact that the fixed XOR count is halved. This is especially so if we can keep the XOR count of a_i and b_i down by choosing them to be 1 or other very lightweight elements. As an example, the lightest sparse DSI k-MDS matrix we found for $n = 8$ and $k = 6$ has a total XOR count of 31, this is already less than the fixed XOR count of cyclic, Hadamard and LFS matrices which is $(6-1) \cdot 8 = 40$. Therefore when we search for lightweight sparse DSI or LFS k-MDS matrices, if we find sparse DSI k-MDS matrix that has XOR count lesser than the fixed cost of LFS matrices,

we do not have to run our search on LFS matrices in hope for finding lighter matrices.

In comparison with GFS matrix. While it seems that sparse DSI matrix is similar to GFS matrix proposed in [20], sparse DSI matrix has two advantages over GFS matrix. Firstly, GFS matrix only exists for even order while our DSI matrix exists for all sizes, thus we can achieve improvements on some parameters that was not achievable by GFS matrix. Secondly, the technique mentioned in Section 4.3 could not be applied to GFS matrix due to the nature of its construction, thus losing the advantage that LFS and DSI matrices have.

6.2 Optimal Serial-type matrix of order 4

It is natural to wonder if it is possible to achieve even lower fixed cost by considering other serial-type matrix structure. Here, we prove that it is not possible for serial-type matrices of order 4. Before that, we state some lemmas that are useful for our proof.

Lemma 1. *If diffusion matrix of order k is MDS, then for every component u_i of the output vector $u \in [\mathrm{GF}(2^n)]^k$, it is some linear combination of all the input components v_i's.*

Proof. Suppose there is an output component u_y that is a linear combination of all input components except v_x. I.e. $u_y = \sum_{i=1}^{k} a_i \cdot v_i$, where $a_i \in GF(2^n)$ and $a_x = 0$. An input vector with all components zero except v_x nonzero has an output vector with at most $k-1$ nonzero components (since u_y is zero), which contradicts that the diffusion matrix is MDS.

Using Lemma 1, we can have the following necessary condition for a serial-type matrices.

1. A necessary condition for serial-type matrices to be MDS when raised to some power q is that every output components is some linear combination of all the input components after q iterations.

The following is a special case of Proposition 2 for serial-type matrices.

Lemma 2. *For any permutation matrices P, the branch numbers of these two matrices M and $P^{-1}MP$ are the same when raised to some power q.*

Proof. When raised to the power of q, we have M^q and $P^{-1}M^qP$. By Proposition 2, they have the same branch number.

For sparse DSI matrices of order 4 over $GF(2^n)$, there are 2 rows with 2 nonzero components, meaning there is a fixed cost of $2n$ XOR count. To achieve lower fixed cost, one has to consider some serial-type matrix structure with only 1 row with 2 nonzero components, denoted as One XOR Serial (OXS) matrices. In

addition, when raised to the power of $q \leq 8$, OXS matrices have to be potentially MDS[7].

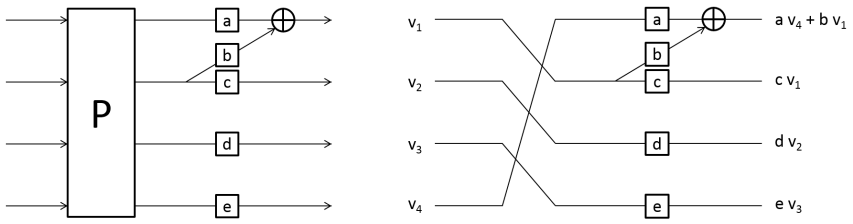
In a nutshell, we want to prove the following theorem:

Theorem 4. *There does not exist OXS matrix of order 4 that is q-MDS, where* $q \leq 8$.

Proof. As seen in Section 4.2, serial-type matrices can be described as some bit permutation followed by an XOR layer. Without loss of generality, we consider the general circuit structure of OXS matrices as in Figure 5a.
Among the 4! possible bit permutations, there are only two permutations that satisfy Condition 1 when $q \leq 8$, namely (1 2 3 4)[8] and (1 2 4 3). Note that they are related by permutation $P = (3\ 4)$, as shown in the following:

$$\begin{pmatrix} b & 0 & 0 & a \\ c & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & e & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} * & 0 & * & 0 \\ * & 0 & 0 & 0 \\ 0 & 0 & 0 & * \\ 0 & * & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

where $*$ is nonzero entry.
By Lemma 2, they have the same branch number. Therefore, we only need to analyse the first permutation, see Figure 5b for the circuit.



(a) General structure of OXS matrix. (b) (1 2 3 4) OXS matrix.

Fig. 5: OXS matrix circuit structure, where P is bit permutation and a, b, c, d, e are field multiplications.

To show that it is not q-MDS where $q \leq 8$, we show that there exists some nonzero input and output vectors pair which has at most 4 nonzero components. The vectors are expressed in terms of the nonzero entries of the OXS matrix.

[7] Given that sparse DSI matrices of order 4 can be 4-MDS, having $q > 8$ would be a bad trade-off between area and clock cycle.

[8] (1 2 3 4) is a cycle permutation expression, where the component in the 1st position goes to 2nd position, 2nd to 3rd, 3rd to 4th, and finally the component in the last position goes to the 1st position.

For $q \leq 7$, consider the input vector $(0, 1, b^{-1}d, 0)^T$. The resultant vectors after each iteration are

$$\begin{pmatrix} 0 \\ 1 \\ b^{-1}d \\ 0 \end{pmatrix} \xrightarrow[i=1]{} \begin{pmatrix} 0 \\ 0 \\ d \\ b^{-1}de \end{pmatrix} \xrightarrow[i=2]{} \begin{pmatrix} ab^{-1}de \\ 0 \\ 0 \\ de \end{pmatrix} \xrightarrow[i=3]{} \begin{pmatrix} 0 \\ * \\ 0 \\ 0 \end{pmatrix} \xrightarrow[i=4]{} \begin{pmatrix} 0 \\ 0 \\ * \\ 0 \end{pmatrix} \xrightarrow[i=5]{} \begin{pmatrix} 0 \\ 0 \\ 0 \\ * \end{pmatrix} \xrightarrow[i=6]{} \begin{pmatrix} * \\ 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow[i=7]{} \begin{pmatrix} * \\ * \\ 0 \\ 0 \end{pmatrix}.$$

By Definition 1, it is not MDS when raised to the power of up to 7.
For $q = 8$, consider the input vector

$$(0, a^{-2}b^{-1}c^{-1}d^{-2}e^{-2} + a^{-3}b^3c^{-2}d^{-3}e^{-3}, a^{-3}b^2c^{-2}d^{-2}e^{-3}, 0)^T.$$

The resultant vectors after each iteration are

$$\begin{pmatrix} 0 \\ a^{-2}(bc)^{-1}(de)^{-2} + a^{-3}b^3c^{-2}(de)^{-3} \\ a^{-3}b^2(cd)^{-2}e^{-3} \\ 0 \end{pmatrix} \xrightarrow[i=1]{} \begin{pmatrix} 0 \\ 0 \\ a^{-2}(bcd)^{-1}e^{-2} + a^{-3}b^3(cd)^{-2}e^{-3} \\ a^{-3}b^2(cde)^{-2} \end{pmatrix}$$

$$\xrightarrow[i=2]{} \begin{pmatrix} a^{-2}b^2(cde)^{-2} \\ 0 \\ 0 \\ a^{-2}(bcde)^{-1} + a^{-3}b^3(cde)^{-2} \end{pmatrix} \xrightarrow[i=3]{} \begin{pmatrix} (abcde)^{-1} \\ a^{-2}b^2c^{-1}(de)^{-2} \\ 0 \\ 0 \end{pmatrix} \xrightarrow[i=4]{} \begin{pmatrix} (acde)^{-1} \\ (abde)^{-1} \\ a^{-2}b^2(cd)^{-1}e^{-2} \\ 0 \end{pmatrix}$$

$$\xrightarrow[i=5]{} \begin{pmatrix} a^{-1}b(cde)^{-1} \\ (ade)^{-1} \\ (abe)^{-1} \\ a^{-2}b^2(cde)^{-1} \end{pmatrix} \xrightarrow[i=6]{} \begin{pmatrix} 0 \\ a^{-1}b(de)^{-1} \\ (ae)^{-1} \\ (ab)^{-1} \end{pmatrix} \xrightarrow[i=7]{} \begin{pmatrix} b^{-1} \\ 0 \\ a^{-1}be^{-1} \\ a^{-1} \end{pmatrix} \xrightarrow[i=8]{} \begin{pmatrix} 0 \\ b^{-1}c \\ 0 \\ a^{-1}b \end{pmatrix}$$

By Definition 1, it is not MDS when raised to the power of 8.

This concludes that our sparse DSI of order 4 has the least fixed XOR count.

7 Conclusion and Future Work

7.1 Conclusion

In this paper, we have proposed a new class of matrices, DSI matrices, and presented several properties and results of these matrices. We also proposed a specific form of DSI matrices, in particular sparse DSI matrices, that have a favourable trade-off of area with throughput (we gain more reduction in hardware area than the increment in the clock cycle).

Using the newly introduced RI property, we can show that the inverse of the serial-type matrices can be of the same cost as the forward direction. This is particularly useful for scenarios where the decryption process is needed as one do not have to pay more for implementing the backward direction of the diffusion matrix.

We presented new lightweight sparse DSI and LFS k-MDS matrices that outperform existing lightweight matrices. Not only do our matrices perform better

in forward direction, we have an advantage in the backward direction where the implementation cost of our inverse matrix is equally lightweight.

Lastly, we proved that for diffusion matrices of order 4, our sparse DSI matrices has the least fixed XOR count, thus closing the search for lower fixed XOR count for matrices of order 4.

7.2 Future work

In the future, we aim to further optimise the search for higher-order sparse DSI k-MDS matrices. It is still an open problem whether such a matrix exists but if it does, we think it will be competitive against existing MDS matrix construction by virtue of having a lower fixed XOR count. Another direction is to search for DSI matrix with invertible binary matrices as entries, which might yield better results.

We also aim to find minimal fixed XOR count for serial-type matrix structure of higher order. We believe that sparse DSI matrices might also be the least fixed cost serial-type matrices for order larger than 4.

References

1. Shannon, C.E.: Communication theory of secrecy systems. Bell system technical journal **28**(4) (1949) 656–715
2. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography. Springer (2002)
3. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis. In: Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000, Proceedings. (2000) 39–56
4. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: CHES. (2011) 326–341
5. Guo, J., Peyrin, T., Poschmann, A. In: The PHOTON Family of Lightweight Hash Functions. Springer Berlin Heidelberg, Berlin, Heidelberg (2011) 222–239
6. Augot, D., Finiasz, M.: Direct construction of recursive MDS diffusion layers using shortened BCH codes. In: Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers. (2014) 3–17
7. Berger, T.P.: Construction of recursive MDS diffusion layers from gabidulin codes. In: Progress in Cryptology - INDOCRYPT 2013 - 14th International Conference on Cryptology in India, Mumbai, India, December 7-10, 2013. Proceedings. (2013) 274–285
8. Gupta, K.C., Pandey, S.K., Venkateswarlu, A.: On the direct construction of recursive MDS matrices. Des. Codes Cryptography **82**(1-2) (2017) 77–94
9. Gupta, K.C., Pandey, S.K., Venkateswarlu, A.: Towards a general construction of recursive MDS diffusion layers. Des. Codes Cryptography **82**(1-2) (2017) 179–195
10. Sim, S.M., Khoo, K., Oggier, F., Peyrin, T.: Lightweight MDS involution matrices. Cryptology ePrint Archive, Report 2015/258 (2015) http://eprint.iacr.org/2015/258.

11. Liu, M., Sim, S.M.: Lightweight MDS generalized circulant matrices (full version). Cryptology ePrint Archive, Report 2016/186 (2016) http://eprint.iacr.org/2016/186.
12. Beierle, C., Kranz, T., Leander, G.: Lightweight multiplication in GF(2^n) with applications to MDS matrices. Cryptology ePrint Archive, Report 2016/119 (2016) http://eprint.iacr.org/2016/119.
13. Li, Y., Wang, M.: On the construction of lightweight circulant involutory MDS matrices. In: Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. (2016) 121–139
14. Vaudenay, S. In: On the need for multipermutations: Cryptanalysis of MD4 and SAFER. Springer Berlin Heidelberg, Berlin, Heidelberg (1995) 286–297
15. H. F. Mattson, J.: The theory of error-correcting codes (F. J. MacWilliams and N. J. A. Sloane). SIAM Review **22**(4) (1980) 513–519
16. Khoo, K., Peyrin, T., Poschmann, A.Y., Yap, H.: FOAM: Searching for hardware-optimal SPN structures and components with a fair comparison. Cryptology ePrint Archive, Report 2014/530 (2014) http://eprint.iacr.org/2014/530.
17. Jean, J., Peyrin, T., Sim, S.M.: Optimizing implementations of lightweight building blocks. Cryptology ePrint Archive, Report 2017/101 (2017) http://eprint.iacr.org/2017/101.
18. Sarkar, S., Syed, H.: Lightweight Diffusion Layer: Importance of Toeplitz Matrices. IACR Trans. Symmetric Cryptol. **2016**(1) (2016) 95–113
19. Gupta, K.C., Ray, I.G.: On constructions of mds matrices from companion matrices for lightweight cryptography. Cryptology ePrint Archive, Report 2013/056 (2013) http://eprint.iacr.org/2013/056.
20. Wu, S., Wang, M., Wu, W.: Recursive diffusion layers for (lightweight) block ciphers and hash functions. In: Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers. (2012) 355–371
21. Sarkar, S., Syed, H., Sadhukhan, R., Mukhopadhyay, D.: Lightweight design choices for led-like block ciphers. In: Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings. (2017) 267–281

A Matrix Examples

Table 5: Sparse DSI and LFS matrix examples. If the matrix has double digit hexadecimal entries, it belongs to GF(2^8)/`0x1c3`. If it has single digit hexadecimal entries, it belongs to GF(2^4)/`0x13`.

Order	Sparse DSI	LFS
4	$D_{4,4} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0x9 & 0 \\ 0 & 0 & 0x2 & 0 \end{pmatrix}, D_{4,8} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0xe1 & 0 \\ 0 & 0 & 0x02 & 0 \end{pmatrix}$	-
5	$D_{5,8} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0x04 & 0 & 0 \\ 0 & 0 & 1 & 0x02 & 0 \\ 0 & 0 & 0 & 0x02 & 0 \end{pmatrix}$	$L_{5,4} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0x1 & 0x2 & 0x9 & 0x9 & 0x2 \end{pmatrix}$
6	$D_{6,8} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0x91 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0x02 & 0 \\ 0 & 0 & 0 & 0 & 0x02 & 0 \end{pmatrix}$	$L_{6,4} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0x1 & 0xd & 0x9 & 0x4 & 0x9 & 0xd \end{pmatrix}$
7	$D_{7,8} = \begin{pmatrix} 0x1c & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0x02 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0x02 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0xb5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0xe1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0xe1 & 0 \end{pmatrix}$	$L_{7,4} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0x1 & 0x2 & 0x7 & 0x1 & 0x1 & 0x7 & 0x2 \end{pmatrix}$
8	-	$L_{8,4} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0x1 & 0xe & 0x2 & 0xd & 0x9 & 0xd & 0x2 & 0xe \end{pmatrix}$

Table 6: Computation of the XOR count using Table 2.

Matrix	Variable XOR	Fixed XOR	Total XOR
$D_{4,4}$	$1+1$	2×4	10
$L_{5,4}$	$1+1$	4×4	18
$L_{6,4}$	$2+1+2$	5×4	25
$L_{7,4}$	$1+5$	6×4	30
$L_{8,4}$	$4+1+2+1$	7×4	36
$D_{4,8}$	$3+3$	2×8	22
$D_{5,8}$	$4+3$	3×8	31
$D_{6,8}$	$3+4$	3×8	31
$D_{7,8}$	$9+3+7+3$	4×8	54

B XOR count

Table 7: XOR Count of elements in GF(2^4)/0x13.

Ele	XOR	Ele	XOR	Ele	XOR	Ele	XOR
0x0	0	0x4	2	0x8	3	0xc	4
0x1	0	0x5	4	0x9	1	0xd	2
0x2	1	0x6	5	0xa	4	0xe	4
0x3	4	0x7	5	0xb	4	0xf	3

Table 8: XOR Count of elements in GF(2^8)/0x1c3. The entries correspond to the XOR count of the element obtained from XORing the corresponding row and column hexadecimal value. For instance, XOR(0x71) = 11.

$\oplus$	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0a	0x0b	0x0c	0x0d	0x0e	0x0f
0x00	0	0	3	8	4	10	9	10	5	11	11	15	11	15	10	13
0x10	7	12	11	13	12	13	14	15	12	16	16	14	9	14	14	17
0x20	9	13	12	14	11	13	13	13	12	14	12	13	13	14	15	14
0x30	14	15	15	17	15	11	13	16	9	14	15	12	14	15	16	11
0x40	10	11	14	16	12	15	15	17	11	15	14	17	13	15	13	16
0x50	13	11	15	15	11	16	13	15	12	14	13	15	14	14	12	9
0x60	16	18	16	16	15	16	16	12	15	15	10	11	13	16	16	11
0x70	8	11	15	14	15	17	13	15	15	14	15	16	16	16	11	11
0x80	12	10	13	15	16	11	16	13	14	15	16	16	16	15	16	16
0x90	11	4	16	11	15	12	15	16	13	14	15	14	12	15	16	15
0xa0	14	10	12	12	17	14	16	16	11	5	16	11	13	13	14	14
0xb0	12	9	16	15	12	7	15	14	14	9	13	9	10	8	8	8
0xc0	16	11	16	14	17	14	16	15	15	12	15	16	15	14	11	16
0xd0	15	15	14	15	11	16	11	16	14	14	15	18	16	16	11	16
0xe0	8	3	11	11	15	11	13	16	14	11	15	14	15	16	16	15
0xf0	15	13	14	13	13	13	15	14	18	16	16	12	11	15	11	16

Note: Values that are larger than 13 are sub-optimal s-XOR values as in [17]. Nevertheless, none of our matrices consists those field elements.