Perfect Diffusion Primitives for Block Ciphers*

Building Efficient MDS Matrices

Pascal Junod and Serge Vaudenay

École Polytechnique Fédérale de Lausanne (Switzerland)
{pascal.junod, serge.vaudenay}@epfl.ch

Abstract. Although linear perfect diffusion primitives, i.e. MDS matrices, are widely used in block ciphers, e.g. AES, very little systematic work has been done on how to find "efficient" ones. In this paper we attempt to do so by considering software implementations on various platforms. These considerations lead to interesting combinatorial problems: how to maximize the number of occurrences of 1 in those matrices, and how to minimize the number of pairwise different entries. We investigate these problems and construct efficient 4×4 and 8×8 MDS matrices to be used e.g. in block ciphers.

1 Introduction

Block ciphers are cascades of diffusion and confusion layers [9]. We usually formalize confusion layers as application of substitution boxes which are defined by lookup tables. Since those tables must be as small as possible for implementation reasons, confusion layers apply substitution in parallel on pieces of informations, e.g. elements whose values lie in a set $\mathcal{K}$ of size 256. The goal of diffusion is to mix up those pieces. One possibility for formalizing the notion of perfect diffusion is the concept of *multipermutation* which was introduced in [8, 10]. By definition, a diffusion function f from $\mathcal{K}^p$ to $\mathcal{K}^q$ is a multipermutation if for any $x_1, \ldots, x_p \in \mathcal{K}$ and any integer r such that $1 \leq r \leq p$, the influence of modifying r input values on $f(x_1, \ldots, x_p)$ is to modify at least $q - r + 1$ output values. Another way to define it consists of saying that the set of all words consisting of $x_1, \ldots, x_p$ concatenated with $f(x_1, \ldots, x_p)$ is a code of $(\#\mathcal{K})^p$ words of length $p + q$ with minimal distance[1] $q + 1$. This notion matches the Singleton bound which relates to MDS codes. Indeed, if $\mathcal{K}$ is a finite field, a linear multipermutation is equivalent to an MDS code expressed in a systematic way, i.e. an arbitrary word of length p is encoded by concatenating it with the linear mapping applied to the word. Since this notion of perfect diffusion was introduced, several block ciphers used the so-called "MDS-matrix" primitive, e.g. AES [2,5], Twofish [6,7],

* Reprint from: P. Junod and S. Vaudenay. Perfect diffusion primitives for block ciphers – building efficient MDS matrices. To appear in *Selected Areas in Cryptography 2004: Waterloo, Canada, August 9-10, 2004. Revised papers*, *Lecture Notes in Computer Science*. Springer-Verlag.

[1] Here the notion of distance is the number of different $\mathcal{K}$-entries.

Khazad [1], or FOX [3], to name a few examples. It is furthermore noteworthy that very few MDS codes are known and they are seldom used in practice. In this paper, we will adopt the following definition of a linear multipermutation.

Definition 1. *Let $\mathcal{K}$ be a finite field and p and q be two integers. Let $x \mapsto M \times x$ be a mapping from $\mathcal{K}^p$ to $\mathcal{K}^q$ defined by the $q \times p$ matrix M. We say that it is a linear multipermutation (or an MDS matrix) if the set of all pairs $(x, M \times x)$ is an MDS code, i.e. a linear code of dimension p, length $p+q$ and minimal distance $q+1$.*

The following theorem [4, Theorem 8 (page 321)] is another way to characterize an MDS matrix.

Theorem 1. *A matrix is an MDS matrix if and only if every sub-matrix is non-singular.*

It is very difficult to define what is “an optimal matrix” in terms of implementation performances, since there exists a large number of criteria which are very dependent of the platform. In this paper we investigate the problem of constructing MDS matrices whose implementation is *very efficient* on most low-cost platforms. For this, we isolate a few criteria which seemed important to us, and we derive several optimality results on these criteria. Note that we only considered one direction, which renders somewhat easier the problem of finding good matrices. Their inverses may not be very efficient but this is not important if we use these matrices with self-inverting constructions, like the Feistel or the Lai-Massey schemes.

2 Performances of Linear Multipermutations

We consider linear multipermutations from $\mathcal{K}^p$ to $\mathcal{K}^q$ where $\mathcal{K}$ is a finite field of characteristic 2. Typically, $\mathcal{K}$ is GF(256). We let M denote a matrix of type $q \times p$ whose elements lie in $\mathcal{K}$. We let $M_{i,j}$ denote the element on row i and column j with $1 \leq i \leq q$ and $1 \leq j \leq p$. The multipermutation is simply $x \mapsto y = M \times x$ where x and y column vectors, i.e.

$$y_i = \sum_{j=1}^{p} M_{i,j} x_j \qquad \text{for } i = 1, \ldots, q.$$

We consider several implementation strategies depending on the platform.

2.1 Software Implementation on 32/64-bit Platforms

Modern 32-bit (or 64-bit) microprocessors with large cache memory[2] lead to well-known and quite simple implementation strategies. Indeed, columns of M

[2] By “cache memory”, we mean the fastest available cache memory, i.e. L1 cache. Most modern CPUs have 16 kB available or more (current versions of the Intel Pentium 4, having 8 kB available, are an exception).

can be partitioned into several sub-columns whose size correspond to the word size (or less). We let w denote the size of the words in terms of $\mathcal{K}$ elements. Then all possible multiplications can be precomputed and put in a table. This means that we consider M as a block matrix of type $\lceil q/w \rceil \times p$, y as a block vector of $\lceil q/w \rceil$ elements, and where every block are vectors of w elements of $\mathcal{K}$, except the blocks in the last row which may be smaller if w does not divide q.

For instance, let us consider 32-bit words, the set $\mathcal{K}$ of bytes (i.e. $w = 4$ and $\#K = 256$), and $p = q = 8$. We let $T_{k,j}$ be a table of 256 4-words vectors such that

$$T_{k,j}(u) = \begin{pmatrix} M_{4k-3,j} \cdot u \\ M_{4k-2,j} \cdot u \\ M_{4k-1,j} \cdot u \\ M_{4k,j} \cdot u \end{pmatrix}$$

for all $u \in \mathcal{K}$ and $k = 1, 2$. Then we can compute $y = M \times x$ by computing $v_k = T_{k,1}(x_1) \oplus \cdots \oplus T_{k,p}(x_p)$ for $k = 1, 2$. Then y is simply the concatenation of v_1 and v_2. Using this approach, we can implement the computation of the linear multipermutation by using $\lceil q/w \rceil \times p$ tables of $\#\mathcal{K}$ entries where each entry is of $w \log_2 \#\mathcal{K}$ bits. For typical applications such as $p = q = 8$, $\#\mathcal{K} = 256$ and $w = 4$ or 8, we have tables of $p \times q \times \#\mathcal{K}$ bytes, i.e. 16 kB of tables. This fits in the fast cache memory of nowadays microprocessors. So this means that we can compute y from x by using only $(p-1) \times \lceil q/w \rceil$ XOR operations, i.e. 14 XORs for $w = 4$ or 7 XORs for $w = 8$, and table lookups. With this approach, performances only depends on $p, q, w, \#\mathcal{K}$ and are independent on the structure of M.

2.2 Software Implementation on 8-bit Platforms

Low-cost 8-bit microprocessors cannot afford to use precomputed data with a size of 16 kB: the matrix multiplication has to be computed on-the-fly. Obviously no $M_{i,j}$ elements can be equal to zero, since this would lead to a singular submatrix of type 1×1 in M and thus would contradict Theorem 1; so we really have to implement $p \times q$ operations. For $\#\mathcal{K} = 256$ we cannot even consider all multiplication tables since this would require naively 64 kB of memory. Another solution would be to express each element x of $\mathcal{K}$ as $x = g^i$, where g is a generator of $\mathcal{K}^*$, and to store the precomputed mappings $x \mapsto i$ and $i \mapsto x$ (one needs 512 bytes of memory). Any multiplication in $\mathcal{K}$ can then be computed using 3 table lookups and 1 addition. However, this approach remains costly. Some multiplication tables are quite simple though. For instance the multiplication by 1 — the neutral element in $\mathcal{K}^*$ — is trivial. Since we need to make multiplications by $M_{i,j}$ only, we may need a small number of tables. Our basic approach is to have all multiplication tables by $M_{i,j}$ elements except for the multiplication table by 1. This leads to the following definitions.

Definition 2. *Let $\mathcal{K}^*$ be a set including a distinguished one denoted 1. Let M be a $q \times p$ matrix whose entries lie in $\mathcal{K}^*$.*

1. *We let $v_1(M)$ denote the number of (i, j) pairs such that $M_{i,j}$ is equal to 1. We call it the* number of occurrences of 1.

2. *We let $c(M)$ be the cardinality of $\{M_{i,j}; i = 1, \ldots, q; j = 1, \ldots, p\}$. We call it the* number of entries*.*
3. *If $v_1(M) > 0$ we let $c_1(M) = c(M) - 1$. Otherwise we let $c_1(M) = c(M)$. We call it the* number of nontrivial entries*.*

With this basic implementation approach we need tables of total size $c_1(M) \times \#\mathcal{K}$ entries in $\mathcal{K}$ in order to implement M. The number of operations consists of $(p-1) \times q$ XORs and number of table lookup's which is equal to $c_1(M_{1,.}) + \cdots + c_1(M_{q,.})$ where $M_{i,.}$ denotes the ith row of M. Indeed, for each row we can look at all equal entries, XOR the corresponding x_j element, look up at the appropriate table, and XOR everything. So the number of CPU operations is within the order of $pq - q + qc_1(M)$. Hence the key metrics for this implementation approach are $c_1(M)$ (for the memory complexity) and $v_1(M)$ (for the time complexity). Note that we may save extra multiplication tables using "efficient GF elements". Here are four typical examples.

- With $\mathcal{K} = \mathrm{GF}(256)$ we can represent a polynomial $a_0 + a_1x + \cdots + a_7x^7$ by the bitstring $a_7 \cdots a_1a_0$. The multiplication by the x element can be implemented by a shift by one bit to the left and a conditional XOR with a constant when a carry bit is set[3].
- Similarly, the multiplication by the x^{-1} element can be implemented by a shift by one bit to the right and a conditional XOR with a constant when a carry bit is set.
- If M includes two elements α and $\alpha + 1$, we can omit the multiplication table by $\alpha + 1$. Multiplication by $\alpha + 1$ is performed by one table lookup (a multiplication by α) and a XOR.
- If M includes two elements α and α^2, we can omit the multiplication table by α^2. Multiplication by α^2 is performed by two consecutive table lookup's.

We can also optimize implementations afterward.

3 Bi-Regular Arrays as Candidates for MDS Matrices

In this section we concentrate on making MDS matrices with high v_1 and low c. The following definition introduces *bi-regular arrays* which are useful objects to build MDS matrices.

Definition 3. *Let $\mathcal{K}^*$ be a set including a distinguished one denoted 1.*

1. *We say that a 2×2 array with entries in $\mathcal{K}^*$ is* bi-regular *if at least one row and one column have two different entries.*
2. *We say that a $q \times p$ array with entries in $\mathcal{K}^*$ is* bi-regular *if all 2×2 sub-arrays are bi-regular.*
3. *An array which is not bi-regular is called* bi-singular*.*

[3] With a special care about side-channel attacks.

4. *Two arrays are* equivalent *if we can obtain the second by performing a finite sequence of simple operations on the first one. Simple operations are permutation of rows, columns, transpose, and permutation of $\mathcal{K}^*$ elements for which 1 is a fixed point.*

Note that an MDS matrix is necessary a bi-regular one (otherwise one 2×2 subdeterminant is singular). Equivalence keeps the bi-regularity. Finally, equivalent arrays have the same v_1 and c metrics. So we can first focus on making bi-regular arrays with high v_1 and low c.

Definition 4. *Let $\mathcal{K}^*$ be a set including a distinguished one denoted 1. We let $v_1^{q,p}$ (resp. $c^{q,p}$) be the maximal (resp. minimal) value of $v_1(M)$ (resp. $c(M)$) for a bi-regular array M of type $q \times p$.*

Note that when $\mathcal{K}^*$ has not enough elements for bi-regular arrays to exist, then $v_1^{q,p}$ and $c^{q,p}$ are undefined. Otherwise $v_1^{q,p}$ and $c^{q,p}$ do not depend on $\mathcal{K}^*$ at all.

One approach for constructing MDS matrices with high v_1 and low c_1 is first to construct a bi-regular array, second to assign elements to some non-zero field values until we get an MDS matrix. We can e.g. look at random values until it succeeds or concentrate on efficient GF elements.

3.1 Highest v_1 for Bi-Regular Arrays

Here are easy facts about $v_1^{q,p}$.

1. We have $v_1^{q,p} = v_1^{p,q}$ since we can transpose bi-regular arrays.
2. We have $v_1^{1,p} = p$ for $p \geq 1$.
3. $v_1^{q,p}$ increases with p and q.

Lemma 1. *The following facts hold:*

- $v_1^{2,p} = p + 1$ *for any* $p \geq 1$.
- $v_1^{3,p} = p + 3$ *for any* $p \geq 3$.
- $v_1^{4,4} = 9$, $v_1^{4,5} = 10$, *and* $v_1^{4,p} = p + 6$ *for any* $p \geq 6$.
- $v_1^{5,5} = 12$, $v_1^{5,6} = 13$, $v_1^{5,7} = 14$, $v_1^{5,8} = 17$, $v_1^{5,9} = 18$, *and* $v_1^{5,p} = p + 10$ *for any* $p \geq 10$.

Proof. For the 2 rows case, the $2 \times p$ array

1	1	1	$\cdots$	1
1	a_2	a_3	$\cdots$	a_p

is bi-regular when $1, a_2, \ldots, a_p$ are pairwise different. We cannot have more occurrences for 1, otherwise we must have two different columns whose entries are only 1, which leads to a bi-singular 2×2 sub-array.

For the 3 rows case, if one column has three occurrences of 1, all other columns must have at most one occurrence of 1 which leads to $p + 2$ in total. If no column has three occurrences of 1, we notice that at most three columns can have two occurrences, which leads to the following construction with $p + 3$ occurrences in total.

1	1	a_1	1	1	$\cdots$	1
1	a_1	1	a_2	a_3	$\cdots$	a_{p-2}
a_1	1	1	a_3	a_4	$\cdots$	a_{p-1}

For the 4 rows case, we similarly prove that no column has four occurrences of 1 in optimal solutions. We cannot have two different columns with 3 occurrences of 1 so we easily notice that the constructions below are optimal.

a_1	1	1	1
1	a_1	a_2	1
1	a_2	1	a_2
1	1	a_2	a_1

a_1	1	1	1	1
1	a_1	a_2	1	a_3
1	a_2	1	a_2	a_3
1	1	a_2	a_1	a_3

When we have more than 5 columns we notice that we get better results when we limit the occurrence number to 2 in every column as done in the following construction.

1	1	1	a_1	a_2	a_3	1	1	1	$\cdots$
1	a_3	a_1	1	1	a_2	a_4	a_5	a_6	$\cdots$
a_1	1	a_2	1	a_3	1	a_5	a_6	a_7	$\cdots$
a_2	a_1	1	a_3	1	1	a_6	a_7	a_8	$\cdots$

For the 5 rows case, we similarly prove that having five occurrences of 1 in the same row leads to sub-optimal solutions. Having a single row with four occurrences, four others with two occurrences, and the others with a single occurrence yields $v_1^{5,5} = 12$, $v_1^{5,6} = 13$, and $v_1^{5,7} = 14$. We can have at most two columns with three occurrences and up to four others with two occurrences, all others being limited to a single occurrence. If we keep a single column with three occurrences then we can have up to seven other columns with two occurrences, all others being limited to a single occurrence. This yields $v_1^{5,8} = 17$. Finally, limiting the occurrences number to two is optimal when we have more than 8 columns since we achieve $v_1^{5,9} = 18$, and $v_1^{5,p} = p + 10$ for any $p \geq 10$. $\square$

We could continue the proof further and obtain $v_1^{6,6} = 16$, $v_1^{6,7} = 18$, $v_1^{6,8} = 19$, $v_1^{7,7} = 21$, $v_1^{7,8} = 22$, $v_1^{8,8} = 24$. The optimal solutions with 6 rows consist of the following array. (For 6 or 7 columns, restrict on the first columns.) Blank cells need to be filled with elements other than 1.

1	1			1			1
1		1			1		
1			1			1	
	1	1				1	
	1		1		1		
		1	1	1			

The optimal solutions with 7 rows and 7 or 8 columns, and 8 rows and columns are the first rows and columns of the following array.

1	1	1					1
1			1	1			
1					1	1	
	1		1		1		
	1			1		1	
		1	1			1	
		1		1	1		
			1				1

The following lemma (with $\alpha = 2$) indicates that $v_1^{n,n}$ can be close to $n\sqrt{n}$ since we can put $\sqrt{n}$ occurrences of 1 in every row.

Lemma 2. *If p is a prime power, for any integers $\alpha > 1$ and $q \leq p^{\alpha-1}(p^\alpha - 1)/(p-1)$ we have $v_1^{q,p^\alpha} \geq q \times p$.*

Proof. This comes from the following construction. We let $\mathcal{K} = \mathrm{GF}(p)$ and we consider the affine space $\mathcal{K}^\alpha$. We have $p^{\alpha-1}(p^\alpha - 1)/(p-1)$ straight lines in total each containing exactly p points. We consider that every column corresponds to a point and that every row corresponds to a straight line. We put 1 in cells in which the corresponding point belongs to the straight line. We fill other cells so that it does not introduce bi-singular sub-arrays. Since straight lines intersect to at most one point, we have a bi-regular array. □

The following lemma provides optimal constructions for small p and q.

Lemma 3. *We have $v_1^{q,p} \geq p + 2q - 3$ for any p, q such that $q \leq p$.*

Proof. This lemma comes from the following construction:

a_{p-1}	1	1	1	1	$\cdots$
1	1	a_2	a_3	a_4	$\cdots$
1	a_{p-1}	1	a_2	a_3	$\cdots$
1	a_{p-2}	a_{p-1}	1	a_2	$\cdots$
1	a_{p-3}	a_{p-2}	a_{p-1}	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

□

In summary, Table 1 gives the first values of $v_1^{q,p}$. Underlined numbers are obtained with the construction of Lemma 3.

3.2 Lowest c for Bi-Regular Arrays

Here are easy facts about $c^{q,p}$.

1. We have $c^{q,p} = c^{p,q}$ since we can transpose bi-regular arrays.
2. We have $c^{1,p} = 1$ for $p \geq 1$. Indeed, the $1 \times p$ array

1	1	1	$\cdots$	1

	2	3	4	5	6	7	8
2	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{7}$	$\underline{8}$	$\underline{9}$
3	4	$\underline{6}$	$\underline{7}$	$\underline{8}$	$\underline{9}$	$\underline{10}$	$\underline{11}$
4	5	7	$\underline{9}$	$\underline{10}$	12	13	14
5	6	8	10	$\underline{12}$	$\underline{13}$	$\underline{14}$	17
6	7	9	12	13	16	18	19
7	8	10	13	14	18	21	22
8	9	11	14	17	19	22	24

Table 1. Values of $v_1^{q,p}$.

is bi-regular and we cannot have more occurrences for 1.

3. $c^{q,p}$ increases with p and q.

Let us now demonstrate other results.

Lemma 4. *We have $c^{2,p} = \lceil\sqrt{p}\rceil$ for any integer $p \geq 1$.*

So we deduce that $c^{q,p} \geq \lceil\sqrt{p}\rceil$ for any $q \geq 2$ and any $p \geq 1$.

Proof. Let $s = \lceil\sqrt{p}\rceil$. Let $a_0, \ldots, a_{s-1}$ be pairwise different. For any $j = 1, \ldots, p$ we first let $j - 1 = qs + r$ be the Euclidean division of $j - 1$ by s, i.e. $0 \leq r < s$. Note that $0 \leq q < s$. We set $M_{1,j} = a_q$ and $M_{2,j} = a_r$. We notice that M is a $2 \times p$ bi-regular array. We have $v_1(M) = s$ and $c(M) = s$. Hence $c^{2,p} \leq s$.

Given an arbitrary $2 \times p$ bi-regular array, let us assume that there are no more than $s - 1$ pairwise different elements in the first row. Since we have more than $s(s-1)$ columns, one element at least occurs at least s times. Let us extract a $2 \times s$ sub array whose first row is a constant element. Note that this sub-array must be bi-regular as well. Obviously the second row must have pairwise different elements. So there are at least s pairwise different elements in the array. Hence $c^{2,p} \geq s$. □

Lemma 5. *For any $k > 1$ we have $c^{k^2-k+1,k!+2} > k$.*

As an application we deduce that $c^{3,4} > 2$ and $c^{7,8} > 3$.

Proof. Let M be a $(k^2 - k + 1) \times (k! + 2)$ array of k elements. We notice that the first column must have an element a with k occurrences. Let us extract the $k \times (k! + 2)$ sub-array M' corresponding to these occurrences. All elements in the first column of M' are equal. If M was bi-regular, M' would be bi-regular as well, so no other column could have two occurrences of the same element b. Hence all columns but the first one would be permutations of the set of elements. Since there are k elements and $k!$ permutations, then two of the other columns would be equal which would contradict the bi-regular property. □

Lemma 6. *For any k we have $c^{2k-1,2k-1} \leq k$.*

As an application we deduce that $c^{3,3} \leq 2$, $c^{5,5} \leq 3$, $c^{7,7} \leq 4$, and $c^{9,9} \leq 5$.

Proof. We construct a bi-regular $(2k-1)\times(2k-1)$ array by using matrices. Let A^r be the $(2k-1)\times(2k-1)$ matrix with integral elements defined by

$$A^r_{i,j} = \begin{cases} 1 \text{ if } |i+j-2k| = 2k-r-1 \text{ or } |i-j| = r \\ 0 \text{ otherwise.} \end{cases}$$

Note that the (i,j) coordinates which lead to $A^r_{i,j} = 1$ lie in a rectangle whose edges are parallel to the diagonals of the matrix and whose (virtual) corners have coordinate

$$\left(\frac{1}{2}, r+\frac{1}{2}\right), \left(2k-r-\frac{1}{2}, 2k-\frac{1}{2}\right), \left(2k-\frac{1}{2}, 2k-r-\frac{1}{2}\right), \left(r+\frac{1}{2}, \frac{1}{2}\right).$$

Finally we let

$$M = a_1A^0 + a_2A^2 + \cdots + a_kA^{2k-2}$$

with pairwise different $a_1, \ldots, a_k$. As examples, here are the 5×5 and 9×9 arrays obtained with $k=3$ and $k=5$.

a	b	b	c	c
b	a	c	b	c
b	c	a	c	b
c	b	c	a	b
c	c	b	b	a

a	b	b	c	c	d	d	e	e
b	a	c	b	d	c	e	d	e
b	c	a	d	b	e	c	e	d
c	b	d	a	e	b	e	c	d
c	d	b	e	a	e	b	d	c
d	c	e	b	e	a	d	b	c
d	e	c	e	b	d	a	c	b
e	d	e	c	d	b	c	a	b
e	e	d	d	c	c	b	b	a

We first notice that for any $1 \le i, j \le 2k-1$ there exists a single even r such that $A^r_{i,j} = 1$. Thus for every cell in M there exists one and only one A^r matrix with r even with the corresponding cell containing 1.

Second we consider a 2×2 sub-array corresponding to positions (i,j), (i,j'), (i',j) and (i',j'). We assume that $M_{i,j} = M_{i,j'}$ and $M_{i',j} = M_{i',j'}$ and we want to lead to a contradiction.

If $i \equiv j \equiv j' \pmod 2$ then $|i-j| = |i-j'|$. Since $j \neq j'$ we deduce $j+j' = 2i$. If $i \not\equiv j \equiv j' \pmod 2$ then $|i+j-2k| = |i+j'-2k|$. Since $j \neq j'$ we deduce $j+j' = 4k-2i$. Since $i \le 2k-1$, we obtain that $j \equiv j' \pmod 2$ implies $i = k - |(j+j')/2 - k|$. The same holds for i'. Since $i \neq i'$ we have a contradiction for the $j \equiv j' \pmod 2$ case.

If $i \equiv j \not\equiv j' \pmod 2$ then $|i-j| = 2k-1-|i+j'-2k|$. So we have $2i = j-j'-1$. Similarly, if $i \not\equiv j \not\equiv j' \pmod 2$ then $2i = j'-j-1$ thus if $j \not\equiv j' \pmod 2$ we have $2i = |j-j'|-1$. The same holds for i'. Since $i \neq i'$ we have a contradiction for the $j \not\equiv j' \pmod 2$ case as well.

So we cannot have $M_{i,j} = M_{i,j'}$ and $M_{i',j} = M_{i',j'}$. By using the transpose, we cannot have $M_{i,j} = M_{i',j}$ and $M_{i,j'} = M_{i',j'}$. So M is bi-regular and $c(M) = k$. □

Lemma 7. *We have* $c^{4,6} \ge 4$.

Proof. The proof can be found in the Appendix.

Lemma 8. *Let q be a prime power. We have $c^{q,q^2-q+1} \leq q$.*

As an application we deduce that $c^{3,7} \leq 3$, $c^{4,13} \leq 4$.

Proof. Let $\mathcal{K}$ be a finite field of cardinality q. We let f be a bijective mapping from $\{2, \ldots, q^2 - q + 1\}$ to $\mathcal{K}^* \times \mathcal{K}$. We let $f(i) = (a_i, b_i)$ for $i = 2, \ldots, q^2 - q + 1$. We let $x_1, \ldots, x_q$ be a numbering of all $\mathcal{K}$ elements. We define $M_{i,1} = 1$ and $M_{i,j} = a_i x_j + b_i$ for $i = 1, \ldots, q$ and $j = 2, \ldots, q^2 - q + 1$. Obviously M is a $q \times (q^2 - q + 1)$ array of q elements. As an example, here is the array with $q = 3$.

a	a	a	b	b	c	c
a	b	c	c	a	a	b
a	c	b	a	c	b	a

Since $a_i \neq 0$ the $x \mapsto a_i x + b_j$ mappings are permutations so all 2×2 sub-array containing the first column are bi-regular. Let us now consider a 2×2 sub-array containing columns j and j' such that $1 < j < j'$. Assuming that $a_i x + b_i = a_{i'} x + b_{i'}$ and $a_i y + b_i = a_{i'} y + b_{i'}$ we have $(a_i - a_{i'})(x - y) = 0$. Since $(a_i, b_i) \neq (a_{i'}, b_{i'})$ we must have $x = y$. Hence the sub-array is bi-regular. □

Lemma 9. *We have $c^{3,8} \geq 4$.*

Proof. Here we must have at least one column which is not a permutation of (abc). Let us assume without loss of generality that the first column is (aax). Then for every other column the entries at row 1 and 2 must be different. But there are only 6 possibilities which is not enough to fill all columns. □

Lemma 10. *We have $c^{6,8} \geq 5$.*

Proof. Assuming that we have a 6×8 array with $c \leq 4$ then for every column we can produce at least two different pairs $\{i, j\}$ corresponding to two equal elements in row i and row j. If the array were bi-regular all pairs would be pairwise different so we would have 16 pairs in total. But we have only $\binom{6}{2} = 15$ possible pairs in total so this is impossible. □

In summary Table 2 provides the obtained $c^{q,p}$ values. Underlined numbers are obtained from Lemma 4, 5, 6, 7, 8, 9, and 10. Other value come from basic properties such as symmetry and monotonicity. The missing element $c^{5,8} \leq 4$ result is obtained by the following construction.

a	a	a	a	d	d	b	b
a	d	c	b	b	a	a	d
b	a	d	c	b	c	d	c
c	b	a	d	a	b	d	a
d	c	b	a	c	b	c	d

	2	3	4	5	6	7	8
2	$\underline{2}$	2	$\underline{2}$	$\underline{3}$	3	3	$\underline{3}$
3	2	$\underline{2}$	$\underline{3}$	3	3	$\underline{3}$	$\underline{4}$
4	2	3	3	3	$\underline{4}$	4	$\underline{4}$
5	3	3	3	$\underline{3}$	4	4	4
6	3	3	4	4	4	4	$\underline{5}$
7	3	3	4	4	4	$\underline{4}$	5
8	3	4	4	4	5	5	$\underline{5}$

Table 2. Values of $c^{q,p}$.

4 MDS Matrices Constructions for $p = q = 4$

We study constructions with $p = q = 4$ over the field $\mathcal{K} = \mathrm{GF}(256)$. Elements are represented as polynomials of degree at most 7 over GF(2). The $a_0 + a_1x + \cdots + a_7x^7$ polynomial is represented by the bitstring $a_7 \cdots a_1a_0$. Formally, x represents a root of an irreducible polynomial of degree 8.

4.1 The AES Matrix

Here is the MDS matrix[4] taken from AES [2,5] with $a = x$ and $b = x + 1$:

$$\begin{pmatrix} a & b & 1 & 1 \\ 1 & a & b & 1 \\ 1 & 1 & a & b \\ b & 1 & 1 & a \end{pmatrix} \tag{1}$$

Multiplication by a is a shift and a conditional XOR. In this case, $c = 3$ is optimal according to our criteria, but $v_1 = 8$ is not. As described in [2], a multiplication by (1) can be implemented (in a pseudo-C notation) using 15 XORs, 4 table lookups and 3 temporary variables:

```
t = a[0] ^ a[1] ^ a[2] ^ a[3]; /* a is the input vector */
u = a[0];
v = a[0] ^ a[1]; v = time[v]; a[0] = a[0] ^ v ^ t;
v = a[1] ^ a[2]; v = time[v]; a[1] = a[1] ^ v ^ t;
v = a[2] ^ a[3]; v = time[v]; a[2] = a[2] ^ v ^ t;
v = a[3] ^ u;    v = time[v]; a[3] = a[3] ^ v ^ t;
```

Note that AES also requires to implement the inverse MDS matrix.

[4] In order to check that this is indeed an MDS matrix, we compute all sub-determinants. They can be expressed as polynomials in terms of x. We can check that none of these polynomials is zero. Since they are all of degree at most 4 and that x is of degree 8, they cannot vanish so we have an MDS matrix.

4.2 An Efficient Matrix

As we have seen, $v_1^{4,4} = 9$ and $c^{4,4} = 3$ and we can hit both optimal criteria with the array of Lemma 3 (M_1 in (2)); let us furthermore consider a second matrix M_2, which is a permuted version of M_1.

$$M_1 = \begin{pmatrix} a & 1 & 1 & 1 \\ 1 & 1 & b & a \\ 1 & a & 1 & b \\ 1 & b & a & 1 \end{pmatrix} \qquad M_2 = \begin{pmatrix} a & 1 & 1 & 1 \\ 1 & a & 1 & b \\ 1 & b & a & 1 \\ 1 & 1 & b & a \end{pmatrix} \tag{2}$$

One can easily verify that necessary conditions for M_2 being a MDS matrix are, for any $a \neq b$ which are not equal to 0 or 1, $a \neq b^2$, $a \neq b + 1$, and $a^2 \neq b$. If we dispose of two multiplication tables (namely, by $a + 1$ and by $b + 1$), we can implement a multiplication by M_2 in the following way:

```
u    = a[0] ^ a[1] ^ a[2] ^ a[3]; /* a is the input vector */
a[0] = u ^ timeap1[a[0]];      v    = timeap1[a[1]];
a[2] = timeap1[a[2]];          a[3] = timeap1[a[3]];
a[1] = u ^ v ^ timebp1[a[3]]; a[3] = u ^ a[3] ^ timebp1[a[2]];
a[2] = u ^ a[2] ^ timebp1[v];
```

This implementation needs 10 XORs, 2 temporary variables, 7 table lookups in two tables. This allows us to decrease the overall number of temporary variables and of operations (at the cost of a supplementary precomputed table), if the XOR operations and table lookups generate identical costs. Note that the same matrix (up to a permutation) forms the diffusive block of FOX64 [3].

5 MDS Matrices Constructions for $p = q = 8$

Here, we give explicit constructions with $p = q = 8$ over $\mathcal{K} = \mathrm{GF}(256)$.

5.1 Circulating-Like Matrix

By using the construction of Lemma 3 with $p = q = 8$, we obtain $v_1 = 21$ and $c = 7$ which, are not optimal. Many different possibilities for filling the coefficients exist; we give here as illustration two different examples.

$$\begin{pmatrix} f & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & a & b & c & d & e & f \\ 1 & f & 1 & a & b & c & d & e \\ 1 & e & f & 1 & a & b & c & d \\ 1 & d & e & f & 1 & a & b & c \\ 1 & c & d & e & f & 1 & a & b \\ 1 & b & c & d & e & f & 1 & a \\ 1 & a & b & c & d & e & f & 1 \end{pmatrix}$$

For GF(256) represented by the irreducible polynomial $x^8+x^4+x^3+x^2+1$ over GF(2), a possible combination is given by $a = x+1$, $b = x^3+1$, $c = x^3+x^2$, $d = x$, $e = x^2$ and $f = x^4$. Note that we need a single precomputed table, namely the multiplication by x. If we can afford two precomputed multiplication tables (by x and by x^{-1}, in this case), when using $x^8+x^7+x^6+x^5+x^4+x^3+1$ as field representation, another possible combination is $a = x+1$, $b = x^{-1}+x^{-2}$, $c = x$, $d = x^2$, $e = x^{-1}$ and $f = x^{-2}$. An implementation using 29 table lookups, 71 XORs is given in Appendix. Note that the same matrix (up to a permutation) forms the diffusive block of FOX128 [3].

5.2 Matrix with Rectangle Patterns

We use the construction of Lemma 6 with $k = 5$ and we remove the first row and the last column. We obtain $v_1 = 15$ and $c = 5$ so this is optimal for c.

$$\begin{pmatrix} b & a & c & b & d & c & 1 & d \\ b & c & a & d & b & 1 & c & 1 \\ c & b & d & a & 1 & b & 1 & c \\ c & d & b & 1 & a & 1 & b & d \\ d & c & 1 & b & 1 & a & d & b \\ d & 1 & c & 1 & b & d & a & c \\ 1 & d & 1 & c & d & b & c & a \\ 1 & 1 & d & d & c & c & b & b \end{pmatrix}$$

Representing GF(256) with $x^8+x^7+x^6+x^5+x^4+x^3+1$ as irreducible polynomial, a possible combination is given by $a = x^{-3}+x^{-1}$, $b = x^{-2}+x^{-1}+1$, $c = x^4+x$ and $d = x$. With $x^8+x^4+x^3+x^2+1$ as irreducible polynomial, a valid combination is $a = x+1$, $b = x^4+1$, $c = x^4+x$ and $d = x$. Using these coefficients, we are able to implement this matrix multiplication with the same amount of table lookups (i.e. 16), 54 XORs instead of 56 and two less temporary variables than the matrix used by the designers of Khazad (as described in [1]), for instance. We might do even better by dedicated optimizations.

6 Conclusion

MDS matrices are a well-known way to build linear multipermutations, i.e. optimal diffusion components which can be used as building blocks of cryptographic primitives, like block ciphers and hash functions. Although their implementation is quite straightforward on 32/64-bit architectures, which have large data L1 caches and thus allow to store large precomputed tables, we need to evaluate the matrix multiplication on-the-fly on low-cost 8-bit architectures, and we can afford only a very limited amount of precomputed data. In this paper, we have studied MDS matrices under the angle of efficiency, defined mathematical criteria and proven several optimality results relatively to these criteria; furthermore, we give new constructions of efficient 4×4 and 8×8 matrices over GF(256).

Future potential investigations may go in the direction of hardware implementations of linear multipermutations, which are not covered by this paper. Furthermore, we may extend our mathematical considerations with criteria specifically dedicated to SPNs; such matrices must have inverses which are also efficient, for fast decryption operations. Finally, we studied bi-regularity of matrices as a necessary condition for being MDS. It is however not sufficient. We indeed have found optimal bi-regular arrays but no instances which are MDS. This problem is left as future work.

Acknowledgments The work presented in this paper was initiated by a project supported by MediaCrypt AG, and supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

1. P. Barreto and V. Rijmen. The Khazad legacy-level block cipher. First Open NESSIE Workshop, Leuven, 2000. See https://www.cryptonessie.org.
2. J. Daemen and V. Rijmen. *The Design of Rijndael.* Information Security and Cryptography. Springer, 2002.
3. P. Junod and S. Vaudenay. FOX: a new family of block ciphers. In *Proceedings of SAC'04.* Springer-Verlag, 2004.
4. F. MacWilliams and N. Sloane. *The theory of error-correcting codes.* North-Holland, 1977.
5. National Institute of Standards and Technology, U. S. Department of Commerce. *Advanced Encryption Standard (AES)*, 2001.
6. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Twofish: A 128-bit block cipher. In *The First AES Candidate Conference.* National Institute for Standards and Technology, 1998.
7. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. *The Twofish encryption algorithm.* Wiley, 1999.
8. C. Schnorr and S. Vaudenay. Black box cryptanalysis of hash networks based on multipermutations. In A. De Santis, editor, *Advances in Cryptology - EUROCRYPT '94. Proceedings*, volume 950 of *LNCS*, pages 47–57. Springer-Verlag, 1995.
9. C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4), 1949.
10. S. Vaudenay. On the need for multipermutations: cryptanalysis of MD4 and SAFER. In B. Preneel, editor, *Fast Software Encryption. Proceedings*, volume 1008 of *LNCS*, pages 286–297. Springer-Verlag, 1995.

A Proof of Lemma 7

First we demonstrate that a 4×6 bi-regular array such that $c = 3$ has no column equivalent to the pattern $(aabb)$. Indeed, let M be a 4×6 array of 3 elements a, b, and c whose first column is $(aabb)$. If the first row has two other occurrences of

a and two occurrences of another element x, we can permute columns in order to get a first row equal to $(aaaxx\cdot)$. Then $M_{2,2}$ and $M_{2,3}$ must be pairwise different, and different from a for M to be bi-regular. Similarly, either $M_{2,4}$ or $M_{2,5}$ must be equal to a. We may permute columns 2 and 3, and columns 4 and 5 and obtain the array below.

a	a	a	x	x	
a	b	c	a		
b	?	?	?		
b	?	?	?		

Positions with question mark cannot be equal to b, so we must fill them with a and c elements. We have three pairs of question marks. Since we only have two elements, either two different pairs are equal, or one pair consists of the same element twice. In both case we contradict the bi-regular property. This means that row 1 cannot be equivalent to $(aaaxx\cdot)$. Obviously row 1 cannot be equivalent to $(aaaa\cdot\cdot)$ (otherwise we have not enough elements to put in row 2 below the a occurrences). For similar reasons row 1 cannot be equivalent to $(axxx\cdot\cdot)$. Thus row 1 must be equivalent to $(aabbcc)$. Since the same arguments hold for row 2, both rows are equivalents. Now let us assume that row 1 is $(aabbcc)$. Looking at what we can put in row 2 we obtain (after potential column permutations) the following array.

a	a	b	b	c	c
a	?	a	c	a	b
b					
b					

So row 2 cannot be equivalent to $(aabbcc)$ which leads to a contradiction. Hence no column can be equivalent to $(aabb)$ in a 4×6 bi-regular array of three elements.

Second, we show that no column can be equivalent to $(xxxy)$. Indeed, if the first column is $(xxxy)$, the elements in row 1, 2, and 3 must be pairwise different in every other column, which leads to 6 possibilities. Let us assume without loss of generality that the array is

x	a	a	b	b	c
x	b	c	a	c	a
x	c	b	c	a	b
y	?				

If the entry at the position of the question mark is b, then the entry at position $(4,3)$ must be different from b and different from the entry at position $(2,3)$, i.e. it must be a. Similarly, if the entry at the position of the question mark is c, the entry at position $(4,3)$ must also be a. After an eventual permutation of column 2 and 3 we can assume that the entry at the position of the question mark is a. But then entries at position $(4,4)$ and $(4,5)$ must be c and a respectively which lead to a singular sub-array.

In conclusion all column must be equivalent to $(xxyz)$. Let us assume that we have the following shape.

x	x'	x''	?		
x	y'	y''	?		
y	x'	z''	?		
z	z'	x''	?		

Then all entries in column 4 must be pairwise different, which is impossible.

B Implementation of the Circulant Matrix

The input is in x[0..7], and the output in y[0..7]. We use two precomputed tables, namely `xtime`[.] (multiplication by x) and `xm1time`[.] (division by x).

```
y[0] = x[0]^x[1]^x[2]^x[3]^x[4]^x[5]^x[6]^xtime[x[7]];
y[1] = x[1]^x[0]^x[7]^xtime[x[1]^x[3]^xtime[4]]^
       xm1time[x[2]^x[5]^xm1time[x[2]^x[6]];
y[2] = x[0]^x[6]^x[7]^xtime[x[0]^x[2]^xtime[3]]^
       xm1time[x[1]^x[4]^xm1time[x[1]^x[5]];
y[3] = x[6]^x[5]^x[7]^xtime[x[6]^x[1]^xtime[2]]^
       xm1time[x[0]^x[3]^xm1time[x[0]^x[4]];
y[4] = x[5]^x[4]^x[7]^xtime[x[5]^x[0]^xtime[1]]^
       xm1time[x[6]^x[2]^xm1time[x[6]^x[3]];
y[5] = x[4]^x[3]^x[7]^xtime[x[4]^x[6]^xtime[0]]^
       xm1time[x[5]^x[1]^xm1time[x[5]^x[2]];
y[6] = x[3]^x[2]^x[7]^xtime[x[3]^x[5]^xtime[6]]^
       xm1time[x[4]^x[0]^xm1time[x[4]^x[1]];
y[7] = x[2]^x[1]^x[7]^xtime[x[2]^x[4]^xtime[5]]^
       xm1time[x[3]^x[6]^xm1time[x[3]^x[0]];
```

C Implementation of the Matrix with Rectangle Patterns

The input is in x[0..7], and the output in y[0..7]. We use two precomputed tables, namely `xtime`[.] (multiplication by x) and `x4time`[.] (multiplication by x^4).

```
t0 = x[0]^x[1]; t1 = x[0]^x[2]; t2 = x[3]^x[5];
t3 = x[1]^x[4]; t4 = x[2]^x[4]; t5 = x[5]^x[7];
t6 = x[3]^x[6]; t7 = x[4]^x[6];
r1 = t1^t5; r2 = t2^t4; r3 = t3^t6; r4 = t2^t6;
y[0] = t0^t6^xtime[t3^t5^x[2]]^x4time[t1^t2];
y[1] = r1^x[4]^xtime[t6^x[1]^x[2]]^x4time[t0^t7];
y[2] = r4^t3^xtime[r1^t2]^x4time[t0^t5];
y[3] = r2^x[6]^xtime[t0^x[4]^x[7]]^x4time[t1^x[6]];
y[4] = r2^x[7]^xtime[t1^x[5]^ x[6]]^x4time[x[2]^x[3]^x[7]];
y[5] = r3^xtime[r1^x[7]]^x4time[t4^x[7]];
y[6] = r1^xtime[r3^x[7]]^x4time[r4];
y[7] = t0^x[6]^x[7]^xtime[r2]^x4time[t5^t7];
```