Double-Matrix: Complete Diffusion in a Single Round with (small) MDS Matrices

Jorge Nakahara Jr

Abstract. This paper describes a simple idea to improve (text) diffusion in block ciphers that use MDS codes but take more than a single round to achieve full (text) diffusion. The Rijndael cipher family is used as an example since it comprises ciphers with different state sizes. A drawback of the new approach is the additional computational cost, but it is competitive compared to large MDS matrices used in the Khazad and Kuznyechik ciphers. A major drawback is that the branch number is not as high as expected compared to those of the Khazad and Kuznyechik ciphers.

Keywords: efficient diffusion, Maximum Distance Separable codes, block ciphers, SPN designs.

1 Introduction

Diffusion is one of C. Shannon's principles [11] for designing secure symmetric-key cryptographic algorithms. Achieving fast diffusion is modern cipher designs means the use of Maximum Distance Separable (MDS) codes [8], which was pioneered in the works by Vaudenay [12], Rijmen *et al.* [9], Daemen and Rijmen [4].

In the Rijndael block cipher family [5, 6], a single 4×4 MDS matrix operating in GF(2^8) is used for all cipher members, although the state size ranges from 16 to 32 bytes organized as 4×4 up to 4×8 cipher states.

This MDS matrix (1) is the redundancy check part of a generator matrix of an $[8, 4, 5]$ MDS code [8] and is used in the MixColumns operation in the finite filed GF(2^8) under the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. The subscript x in (1) means hexadecimal notation.

$$\begin{bmatrix} 02_x & 03_x & 01_x & 01_x \\ 01_x & 02_x & 03_x & 01_x \\ 01_x & 01_x & 02_x & 03_x \\ 03_x & 01_x & 01_x & 02_x \end{bmatrix} \tag{1}$$

Complete text diffusion[1] is Rijndael takes two or more rounds depending on the block size [6].

Other block cipher designs, such as Khazad [3] and Kuznyechik [1], achieve full text diffusion in a single round by using comparatively larger MDS matrices. In Khazad, the state size consists of 8 bytes while its MDS matrix M is 8×8. Let MC denote the MDS matrix transformation in Khazad, and let $X \in (\mathrm{GF}(2^8))^8$ denote the cipher state as a 1×8 row vector. Then,

$$MC(X) = X * M \tag{2}$$

[1] For this paper, complete or full (text) diffusion means that each byte of the output cipher state depends on all bytes of the input cipher state.

or

$$MC(X) = M * X^T \tag{3}$$

where X^T denotes a 8×1 column vector. In a single matrix multiplication, according to (2) or (3), each byte of MC(X) depends linearly on each byte of X.

Likewise for the Kuznyechik cipher where the state X consists of 16 bytes (as a row or column vector) and its MDS matrix M in 16×16.

The faster diffusion in Khazad and Kuznyechik is offset by a higher computational cost per round compared to the cost per round in the Rijndael cipher family which uses a 4×4 MDS matrix.

Let the round operations in Rijndael be denoted by AK_i for the AddRoundKey operation of the i-th round, MC for MixColumns, SR for ShiftRows, and SB for SubBytes. Further details for each operation can be found in [6]. A full round in Rijndael will be denoted ρ_i, and defined as

$$\rho_i(X) = \mathrm{AK}_i \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}(X) = \mathrm{AK}_i(\mathrm{MC}(\mathrm{SR}(\mathrm{SB}(X)))),$$

where[2] X is the $4 \times m$ internal cipher state with $4 \leq m \leq 8$.

A simple idea we explore to achieve full (text) diffusion in a single round of an SPN block cipher design with an $n \times m$ state such as the Rijndael cipher family, is to use two small MDS matrices of orders n and m, respectively, one multiplied on the left-hand side and the other on the right-hand side of the state. So, instead of the round function ρ_i, we have the modified round function

$$\rho'_i(X) = \mathrm{AK}_i(\mathrm{MC'}(\mathrm{SB}(X)))$$

where

$$\mathrm{MC'}(X) = M_1 * X * M_2$$

with M_1 a 4×4 MDS matrix, M_2 a $m \times m$ MDS matrix and $4 \leq m \leq 8$.

Example: consider an MDS matrix M of order $n = m = 2$ and a 2×2 cipher state X with entries (a, b, c, d) organized as in (4). Let the MDS matrix elements be the nonzero tuple (m_1, m_2, m_3, m_4). Multiplying M on the left of X results in the intermediate state X':

$$M * X = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = X' = \begin{bmatrix} a.m_1 + c.m_2 & b.m_1 + d.m_2 \\ a.m_3 + c.m_4 & b.m_3 + d.m_4 \end{bmatrix} \tag{4}$$

where . and + are the multiplication and addition operations in a finite field.

The (text) diffusion is not yet complete since each entry of X' depends on only two (out of four) original state entries. Further, multiplying X' by M on the right-hand side, results in X'':

$$X' * M = \begin{bmatrix} a.m_1 + c.m_2 & b.m_1 + d.m_2 \\ a.m_3 + c.m_4 & b.m_3 + d.m_4 \end{bmatrix} * \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} = X'' =$$

$$\begin{bmatrix} (a.m_1 + c.m_2).m_1 + (b.m_1 + d.m_2).m_3 & (a.m_1 + c.m_2).m_2 + (b.m_1 + d.m_2).m_4 \\ (a.m_3 + c.m_4).m_1 + (b.m_3 + d.m_4).m_3 & (a.m_3 + c.m_4).m_2 + (b.m_3 + d.m_4).m_4 \end{bmatrix}$$

The first matrix multiplication guarantees diffusion per column of X while the second matrix multiplication guarantees diffusion per row of X'. When multiplied on both sides of X, the result X'' depends on all four original inputs (a, b, c, d) of X, that is, complete text diffusion is achieved in a single round assuming both matrix multiplications are performed in the same round.

[2] Note that in Rijndael, the state X is not a row vector nor a column vector, but a rectangular matrix.

If M is not involutory[3], then there will be a performance gap between the encryption and decryption operations since for encryption $M * X * M$ will not cost the same as $M^{-1} * X * M^{-1}$ for decryption. To balance the computational cost for cipher states that are square matrices, there are two alternatives:

(i) use an involutory MDS matrix M, so that the transformation $M * X * M$ is the same (and therefore, costs the same) for both encryption and decryption;
(ii) M is not involutory. In this case, $M * X * M^{-1}$ can be used for encryption and the inverse transformation for decryption will be $M^{-1} * X * M$, which again, will cost the same for both operations.

The approach (i) is appropriate for the Anubis cipher [2], whose matrix is involutory, leading to the same computational cost for both encryption and decryption.

For the AES cipher, whose MDS matrix (1) is not involutory, the approach (ii) leads to the use of (1) and its inverse matrix (5) for both encryption and decryption, making the new combined transformation MC' have the same cost.

$$\begin{bmatrix} 0E_x & 0B_x & 0D_x & 09_x \\ 09_x & 0E_x & 0B_x & 0D_x \\ 0D_x & 09_x & 0E_x & 0B_x \\ 0B_x & 0D_x & 09_x & 0E_x \end{bmatrix} \tag{5}$$

Let M be the AES matrix (1) and let the state matrix X be (6).

$$X = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \tag{6}$$

Let X denote the internal AES state after SubBytes. If we remove ShiftRows and apply (1) on the left of X, the partial state X' is (7):

$$X' = M * X = \begin{bmatrix} 2_x & 3_x & 1_x & 1_x \\ 1_x & 2_x & 3_x & 1_x \\ 1_x & 1_x & 2_x & 3_x \\ 3_x & 1_x & 1_x & 2_x \end{bmatrix} * \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

$$= \begin{bmatrix} 2_x a + 3_x e + i + m & 2_x b + 3_x f + j + n & 2_x c + 3_x g + k + o & 2_x d + 3_x h + l + p \\ a + 2_x e + 3_x i + m & b + 2_x f + 3_x j + n & c + 2_x g + 3_x k + o & d + 2_x h + 3_x l + p \\ a + e + 2_x i + 3_x m & b + f + 2_x j + 3_x n & c + g + 2_x k + 3_x o & d + h + 2_x l + 3_x p \\ 3_x a + e + i + 2_x m & 3_x b + f + j + 2_x n & 3_x c + g + k + 2_x o & 3_x d + h + l + 2_x p \end{bmatrix} \tag{7}$$

The final state after a single round is $X'' = X' * M^{-1} = M * X * M^{-1}$ depicted in (8):

$$X'' = \begin{bmatrix} a'' & b'' & c'' & d'' \\ e'' & f'' & g'' & h'' \\ i'' & j'' & k'' & l'' \\ m'' & n'' & o'' & p'' \end{bmatrix} \tag{8}$$

[3] An involutory matrix M is its own inverse: $M = M^{-1}$.

where, $a'' = (2_x a + 3_x e + i + m).E_x + (2_x b + 3_x f + j + n).9_x + (2_x c + 3_x g + k + o).D_x + (2_x d + 3_x h + l + p).B_x$,
$b'' = (2_x a + 3_x e + i + m).B_x + (2_x b + 3_x f + j + n).E_x + (2_x c + 3_x g + k + o).9_x + (2_x d + 3_x h + l + p).D_x$,
$c'' = (2_x a + 3_x e + i + m).D_x + (2_x b + 3_x f + j + n).B_x + (2_x c + 3_x g + k + o).E_x + (2_x d + 3_x h + l + p).9_x$,
$d'' = (2_x a + 3_x e + i + m).9_x + (2_x b + 3_x f + j + n).D_x + (2_x c + 3_x g + k + o).B_x + (2_x d + 3_x h + l + p).E_x$,
$e'' = (a + 2_x e + 3_x i + m).E_x + (b + 2_x f + 3_x j + n).9_x + (c + 2_x g + 3_x k + o).D_x + (d + 2_x h + 3_x l + p).B_x$,
$f'' = (a + 2_x e + 3_x i + m).B_x + (b + 2_x f + 3_x j + n).E_x + (c + 2_x g + 3_x k + o).9_x + (d + 2_x h + 3_x l + p).D_x$,
$g'' = (a + 2_x e + 3_x i + m).D_x + (b + 2_x f + 3_x j + n).B_x + (c + 2_x g + 3_x k + o).E_x + (d + 2_x h + 3_x l + p).9_x$,
$h'' = (a + 2_x e + 3_x i + m).9_x + (b + 2_x f + 3_x j + n).D_x + (c + 2_x g + 3_x k + o).B_x + (d + 2_x h + 3_x l + p).E_x$,
$i'' = (a + e + 2_x i + 3_x m).E_x + (b + f + 2_x j + 3_x n).9_x + (c + g + 2_x k + 3_x o).D_x + (d + h + 2_x l + 3_x p).B_x$,
$j'' = (a + e + 2_x i + 3_x m).B_x + (b + f + 2_x j + 3_x n).E_x + (c + g + 2_x k + 3_x o).9_x + (d + h + 2_x l + 3_x p).D_x$,
$k'' = (a + e + 2_x i + 3_x m).D_x + (b + f + 2_x j + 3_x n).B_x + (c + g + 2_x k + 3_x o).E_x + (d + h + 2_x l + 3_x p).9_x$,
$l'' = (a + e + 2_x i + 3_x m).9_x + (b + f + 2_x j + 3_x n).D_x + (c + g + 2_x k + 3_x o).B_x + (d + h + 2_x l + 3_x p).E_x$,
$m'' = (3_x a + e + i + 2_x m).E_x + (3_x b + f + j + 2_x n).9_x + (3_x c + g + k + 2_x o).D_x + (3_x d + h + l + 2_x p).B_x$,
$n'' = (3_x a + e + i + 2_x m).B_x + (3_x b + f + j + 2_x n).E_x + (3_x c + g + k + 2_x o).9_x + (3_x d + h + l + 2_x p).D_x$,
$o'' = (3_x a + e + i + 2_x m).D_x + (3_x b + f + j + 2_x n).B_x + (3_x c + g + k + 2_x o).E_x + (3_x d + h + l + 2_x p).9_x$,
$p'' = (3_x a + e + i + 2_x m).9_x + (3_x b + f + j + 2_x n).D_x + (3_x c + g + k + 2_x o).B_x + (3_x d + h + l + 2_x p).E_x$,
and every byte of X'' depends linearly on every byte of the original state X.

So far we considered cipher states as square matrices such as the 4×4 state of AES.

For rectangular states, such as the 24-byte state of Rijndael-192[4], an alternative approach is to use two MDS matrices: a 4×4 matrix M_1 and a 6×6 matrix M_2. This situation is more asymmetric because M_1 and M_2 have different dimensions.

Let X denote the 4×6 cipher state of Rijndael-192. The new diffusion layer applied to Rijndael-192 would remove ShiftRows and consist of $M_1 * X * M_2$ for encryption (where X is the cipher state after SubBytes), and $M_1^{-1} * X * M_2^{-1}$ for decryption. Although this modified Rijndael-192 is more expensive (per round) than the original Rijndael-192, it is wya cheaper than using a (much) larger 24×24 MDS matrix (which is the approach used in Khazad and Kuznyechik).

Since the state of Rijndael-192 is not square but rectangular, it is not possible to use the same matrix on both sides of the internal state X. But, it is still possible to use involutory MDS matrices of orders 4 and 6, so that encryption and decryption share the same computational cost[5].

For Rijndael-256, which uses a 32-byte state, an alternative diffusion layer uses two MDS matrices: a 4×4 matrix M_1 and a 8×8 matrix M_2. Let X denote the 4×8 cipher state of Rijndael-256. The new diffusion layer of Rijndael-256 would remove ShiftRows and consist of $M_1 * X * M_2$ for encryption (where X is the cipher state after SubBytes), and $M_1^{-1} * X * M_2^{-1}$ for decryption. Although it is more expensive per round than Rijndael-256, using M_1 and M_2 is still much cheaper than using a 32×32 MDS matrix to achieve full text diffusion in a single round.

2 Performance

The exact performance loss of any modified Rijndael cipher compared to the original Rijndael design will depend on the exact MDS matrix coefficients and also on the metric used [7].

To get a rough estimate of the computational cost of the MC and MC' layers, we will restrict the analysis to the AES cipher with 128-bit keys, which we denote as AES-128.

[4] The suffix indicates the block size in bits.

[5] We do not provide any particular MDS matrix of order 6 here, but [10] is a very good source of MDS matrices of different orders.

To measure the cost of text diffusion in AES, a software implementation was timed (in CPU cycles) **in ECB mode** in an Intel Core i5 CPU, for a several 128-bit text blocks and keys, **with** and **without** the diffusion components.

The average timings in #CPU cycles are in Tables 1 and 2.

Table 1. Average #CPU cycles for one 128-bit text block AES-128 encryption/decryption.

#CPU cycles per AES-128 encryption	#CPU cycle per AES-128 decryption
5.066100 (in 10000 trials)	4.522200 (in 10000 trials)
3.977120 (in 100000 trials)	4.103070 (in 100000 trials)
3.876024 (in 1000000 trials)	3.970893 (in 1000000 trials)

Table 2. Average #CPU cycles for one 128-bit text block AES-128 encryption/decryption without the MixColumns layer.

#CPU cycles per AES-128 encryption	#CPU cycle per AES-128 decryption
2.977600 (in 10000 trials)	3.130500 (in 10000 trias)
2.920230 (in 100000 trials)	3.111800 (in 100000 trials)
2.985882 (in 1000000 trials)	2.941333 (in 1000000 trials)

From the last lines in Tables 1 and 2, the difference is $3.8760 - 2.9858 = 0.8901$ CPU cycles, which means $0.8901/3.8760 \approx 0.2296$ ie. the MixColumns layer is estimated to cost roughly 23% of each 10-round AES-128 encryption of a 128-bit text block.

Likewise, for AES-128 decryption, the difference is $3.970893 - 2.941333 = 1.02956$ CPU cycles, which means $1.02956/3.970893 \approx 0.2592$ ie, the InvMixColumns layer is estimated to cost roughly 26% of each 10-round AES-128 decryption of a 128-bit text block. This result corroborates the fact that InvMixColumns costs more than MixColumns.

Thus, if we assume the approach (ii) for a modified 10-round AES-128 encryption ie. multiplying the cipher state with matrix (1) on the left and matrix (5) on the right would cost roughly 26% more than the original AES-128.

We leave a more precise cost estimate for future analysis. The same applies to other members of the Rijndael cipher family.

Comparatiively, if we were to multiply the AES-128 state by a 16×16 MDS matrix M_{16} as in the Kuznyechik cipher, we would have to consider that M_{16} can accomodate sixteen 4×4 matrices inside it. We do not go into the details of what the M_{16} entries are, but the computational cost of multiplying a 16-byte state (e.g as a row vector) by M_{16} would be more than sixteen times that of a single 4×4 MDS matrix because M_{16}, being itself MDS, cannot have sixteen identical copies of the same 4×4 MDS matrix. Therefore, M_{16} would cost more than sixteen MixColumns operations.

3 Security

Concerning security against DC and LC, we restrict the analysis to AES, and consider two situations:

(a) the original AES,
(b) a modified AES where there is no ShiftRows transformation and the MixColumns operation is replaced by $M * X * M^{-1}$, where X is the AES state, M is the matrix (1), and M^{-1} is its inverse (5).

In (a), complete text diffusion is achieved after two full rounds, and the sequence of the minimum number of (differentially or linearly) active S-boxes (across five rounds) follows the pattern: 1-4-16-4-1. Therefore, it takes four rounds to reach 25 active S-boxes, which is enough to protect the AES against DC and LC [6].

In (b), complete text diffusion is achieved after one round, but the sequence of minimum number of (differentially or linearly) active S-boxes is not as good as in the long run. Consider the first matrix multiplication in (7). If we activate four bytes, then at least one byte is active in the output because (1) is an MDS matrix. Let us assume there is exactly one active byte. Then, after the second matrix multiplication, the output contains exactly four active bytes. It means that the branch number of (8) is at most 4+4. Thus, if this pattern of active S-boxes repeats then it will take (theoretically) at least six rounds to reach 24 active S-boxes (4+4+4+4+4+4).

Therefore, the security of (b) against DC and LC is worse than the original AES design.

4 Conclusions

This paper proposed a simple idea to achieve full text diffusion in a single round in SPN ciphers using small MDS matrices.

As examples, the Rijndael cipher family was used to illustrate this idea because it has rectangle-shaped states that are convenient to illustrate our arguments and this fact allows us to show the flexibility of our approach to non-square-shaped cipher states.

The advantage of the new approach is a faster (text) diffusion, in a single round, of existing cipher designs, which may help counter attacks that exploit slow diffusion. The drawbacks are: (i) the new design decreases performance, and most critical: (ii) the branch number is worse across consecutive rounds, i.e the modified ciphers will need more rounds to activate enough S-boxes to counter differential and linear attacks[6].

References

1. R. AlTawy, O. Duman, and A.M. Youssef. Fault Analysis of Kuznyechik. IACR ePrint archive, 2005-347, 2005.
2. P. S. L. M. Barreto and V. Rijmen. The ANUBIS Block Cipher. First NESSIE Workshop, Heverlee, Belgium, 2000.
3. P. S. L. M. Barreto and V. Rijmen. The KHAZAD Legacy-Level Block Cipher. First NESSIE Workshop, Heverlee, Belgium, 2000.
4. J. Daemen, L. R. Knudsen, and V. Rijmen. The Block Cipher SQUARE. In E. Biham, editor, *Fast Software Encryption (FSE)*, LNCS 1267, pages 149–165. Springer, 1997.
5. J. Daemen and V. Rijmen. AES Proposal: Rijndael. First AES Conference, California, USA, http://www.nist.gov/aes, 1998.
6. J. Daemen and V. Rijmen. *The Design of Rijndael, AES - The Advanced Encryption Standard*. Springer, 2002.

[6] Many thanks to Vincent Rijmen for insightful discussions on this idea.

7. L. Kolsch. XOR-counts and Lightweight Multiplication with Fixed Elements in Binary Finite Fields. IACR ePrint archive, 2019-229, 2019.
8. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library. North-Holland Publishing Co., 1977.
9. V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De, Win. The Cipher SHARK. In D. Gollmann, editor, *Fast Software Encryption (FSE)*, LNCS 1039, pages 99–112. Springer, 1996.
10. A. Clara Zoppi Serpa. Study of Diffusion and MDS Matrices in Symmetric Block Ciphers. MSc dissertation, University of Campinas, UNICAMP, 2023.
11. C. E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.
12. S. Vaudenay. On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER. In B. Preneel, editor, *Fast Software Encryption (FSE)*, LNCS 1008, pages 286–297. Springer, 1994.