Direct construction of quasi-involutory recursive-like MDS matrices from 2-cyclic codes

Victor Cauchois[1], Pierre Loidreau[1] and Nabil Merkiche[2]

[1] DGA MI and IRMAR, Université de Rennes 1, France
[2] DGA IP and Sorbonnes universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, France[†]

Abstract. A good linear diffusion layer is a prerequisite in the design of block ciphers. Usually it is obtained by combining matrices with optimal diffusion property over the Sbox alphabet. These matrices are constructed either directly using some algebraic properties or by enumerating a search space, testing the optimal diffusion property for every element. For implementation purposes, two types of structures are considered: Structures where all the rows derive from the first row and recursive structures built from powers of companion matrices. In this paper, we propose a direct construction for new recursive-like MDS matrices. We show they are quasi-involutory in the sense that the matrix-vector product with the matrix or with its inverse can be implemented by clocking a same LFSR-like architecture. As a direct construction, performances do not outperform the best constructions found with exhaustive search. However, as a new type of construction, it offers alternatives for MDS matrices design.

Keywords: diffusion layers · MDS matrices · involutions · cyclic codes

1 Introduction

The construction of good linear diffusion layers that can be efficiently implemented both in hardware and in software is an important challenge in the design of block ciphers or hash functions. Since the design of the AES *MixColumn* function, this issue has been thoroughly considered. Namely, these matrices form the ground on which diffusion layers are designed. MDS matrices ensure the maximal diffusion of symbols (usually bytes or nibbles), giving thus a direct lower bound on the bit diffusion.

From coding theory MDS matrices have been known for decades as redundant part of generator matrices of MDS codes under systematic form, [MS77, PH98]. However not all of them can lead to efficient hardware or software implementation. To this end, two types of matrix structure are usually investigated:

- Matrices such that every row is a permutation of the symbols of the first row. The AES *MixColum* matrix is cyclic and hence is typically of this form, [DR02]. In software, the matrix-vector product can be tabulated efficiently. In hardware, the structure of the matrices minimizes the number of symbol field multipliers to implement. In [SKOP15], the authors presented a good survey of lightweight MDS matrices with such properties.
- Serial or recursive matrices. They are powers of companion matrices of polynomials over the symbol field. The matrix-vector products can be implemented with a Feistel structure or by clocking an LFSR. This kind of matrices is well-suited for hardware implementation. It is typically the structure chosen in the design of the PHOTON family of hash functions or the LED block cipher, [GPP11, GPPR11].

[†]This work has been supported in part by ANR ARRAND 15-CE39-0002-01.

There are mainly two methods to construct MDS matrices. The first one consists in enumerating matrices in some search space and testing whether they are MDS or not. It is in particular what was done for the AES *MixColumn*, PHOTON or LED matrices. However, this method only works when the size of the searched matrices remains relatively small since the procedure of testing whether a matrix is MDS requires to compute all the minors of the matrix, which can quickly become computationally intractable.

The second method consists in a direct construction by using tools from coding theory. It has been known since the 80's that some types of matrices such as generalized Cauchy matrices are MDS, [RL89]. Another example of direct construction of MDS matrices consists in taking any subsquare matrix from a Hankel configuration, [RS85, Aid86]. In this set of directly constructed MDS matrices, designers look for matrices with interesting properties for implementation such as being cyclic or of Hadamard type. In [AF14], the authors showed how to construct MDS recursive matrices by shortening an MDS BCH code in an extension field of the symbol alphabet. Berger, in [Ber13], proposed a direct construction of recursive Maximum Rank Distance (MRD[1]) matrices (which implies that these matrices are also MDS) by proving that the redundancy matrix of some Gabidulin codes is a power of the companion matrix of some polynomial.

Sometimes, designers may be interested in additional properties, especially to save resources in hardware implementation. It is the case when the MDS matrix is involutory since the matrix-vector product by the matrix or by its inverse can be implemented with the same circuit. However, this latter requirement imposes a strong constraint on the matrices, which in some cases do not even exist. For instance, Gupta *et al* showed that cyclic involutory MDS matrices do not exist [GR14]. In [SKOP15], the authors proposed search algorithms of lightweight involutory MDS matrices. More recently, Liu *et al* [LS16] relaxed classical circulant matrices definitions to introduce *cyclic matrices* to prove the existence of involutory cyclic MDS matrices. Li *et al* [LW16] relaxed classical MDS definition by considering $m \times m$ non-singular matrices over $\mathbb{F}_2$ instead of field elements in $\mathbb{F}_{2^m}$ to show the existence of matrices with the same symbol diffusion that are both involutory and circulant.

Our Contributions

We propose a new direct construction of MDS matrices. These matrices are obtained as *skewed* products of the companion matrix associated with a polynomial. The *skewed* product is obtained by the use of the squaring function. Thanks to this *skewed* recursive structure, our MDS matrices can be efficiently implemented with a *skewed* linear feedback shift-register (SLFSR) where multiplications by symbols are *skewed* by squaring. This mimics the approach in [AF14], showing that MDS cyclic codes lead to redundancy matrices which can be recursively implemented as powers of companion matrices associated with some polynomials. However, their approach is made more complex by the fact that BCH MDS codes of even length do not exist in fields of characteristic 2.

By introducing the notion of 2-cyclicity which *skews* the notion of cyclicity, we do not have this problem and our results are straightforward consequences of the theory of Gabidulin codes [Gab85], further extended in [BGU07, CLU09, BU09].

Moreover, the MDS matrices that we obtain fulfill the additional *quasi-involutory* condition: The inverse matrix is obtained by squaring the coefficients of the MDS matrix a fixed number of times equal to the size of the matrix. From implementation considerations, the squaring is an automorphism of the symbol field, this means that the inverse matrix-vector product can be implemented by one additional bit permutation under proper implementation settings.

[1]An MRD code is also an optimal code for the rank metric used in network coding and in some asymmetric code-based primitives

Finally, as a direct construction, we do not aim at finding matrices that present better performances than the best known matrices, found with exhaustive search. Still, it offers a new way to construct MDS matrices that could benefit to situations when the complexity of exhaustive search is prohibitive. We also show that the SLFSR structure can by itself lead to MDS matrices comparable in terms of XOR count to the best known matrices validating thus the interest of considering the SLFSR architecture to design MDS matrices.

Outline of the document

In a first part, we present interesting properties of cyclic codes. In characteristic 2, the direct construction of MDS matrices from cyclic codes leads to a dead-end. In section 3, we introduce the notion of 2-cyclicity and compare it with the classical definition of cyclicity, a common tool of coding theory. We show that it is strongly related to a polynomial ring called 2-polynomial ring where arithmetic operations can be efficiently performed. Similarly to the cyclic case this leads to the design of matrices recursively constructed by skewing the product of the companion matrices of the generating 2-polynomial. Additionally, we prove that these matrices have some kind of *quasi-involutory* property. Section 4 deals with 2-cyclic Gabidulin codes. These codes are the equivalent of BCH codes in the ring of 2-polynomials, giving thus a condition on their minimum distance. In well-chosen cases they form a family of MDS 2-cyclic codes. Therefore, they provide us with recursively constructed MDS matrices. In a final section, we show that the direct product with these matrices can be efficiently implemented in hardware under the form of an LFSR skewed by the squaring operator. This operator can be implemented with a bit permutation in a proper basis (normal basis for instance, [Men93]). In this setting, the inverse product is obtained by the same circuit simply by adding another bit permutation, therefore with negligible additional complexity. We also compare the SLFSR implementation cost in terms of XOR count and show that in the case of the field $GF(2^4)$ it is competitive with the best known matrices.

Notation and preliminaries

We denote by $GF(2^{2m})$ the finite field with 2^{2m} elements. In the following, the matrices we will consider are $m \times m$ matrices whose coefficients lie in $GF(2^{2m})$.

Given an MDS-linear code $\mathcal{C}$, the redundant part of a generator matrix of $\mathcal{C}$ under the systematic form is an MDS matrix. In other words, if

$$\mathcal{C} = \langle (R \mid I) \rangle$$

modulo any permutation of the code positions, then R is MDS.

Let $g(X) \in GF(2^{2m})[X]$ be a monic polynomial:

$$g(X) = X^m + \sum_{i=0}^{m-1} g_i X^i$$

The *companion matrix associated with* g is the $m \times m$ matrix we denote by C_g defined:

$$C_g = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ g_0 & g_1 & \cdots & \cdots & g_{m-1} \end{pmatrix} \tag{1}$$

For an element x of $GF(2^{2m})$, we use the notation $x^{[i]} = x^{2^i}$, meaning that x is squared i times. We make an important use of the squaring of the coefficients of matrices.

Therefore, for any matrix $A = (a_{k,l})$, we denote by $A^{[i]} = \left(a_{k,l}^{[i]}\right)$ the matrix obtained by squaring i times all of its coefficients. This is different from the classical power of a matrix A^{2^i} but has the following useful properties:

Proposition 1. *Let $A = (a_{k,l})$ and $B = (b_{k,l})$ with compatible sizes. We have*

- $\left(A^{[i]}\right)^{[j]} = A^{[i+j]}$ *for all integers i and j*
- $(AB)^{[i]} = A^{[i]}B^{[i]}$ *for all integers i*

The proof is direct from the fact that the squaring is a field morphism in any field of characteristic 2: For any elements x and y in $GF(2^{2m})$

- $(x+y)^{[1]} \overset{def}{=} (x+y)^2 = x^2 + y^2 \overset{def}{=} x^{[1]} + y^{[1]}$
- $(xy)^{[1]} \overset{def}{=} (xy)^2 = x^2y^2 \overset{def}{=} x^{[1]}y^{[1]}$

Finally note that if x is an element of $GF(2^{2m})$, then $x^{[2m]} = x$.

2 Cyclic codes and polynomial rings

Definition 1 (cyclic codes)**.** Let $\mathcal{C}$ be a linear code of length $2m$ over $GF(2^{2m})$. Then $\mathcal{C}$ is cyclic if for any vector $\mathbf{x} = (x_0, \ldots, x_{2m-1}) \in \mathcal{C}$, the rotated vector:

$$(\mathbf{x} \ggg 1) = (x_{2m-1}, x_0, \ldots, x_{2m-2})$$

belongs to $\mathcal{C}$.

Cyclic codes are a well-known long-time studied object in coding theory, [MS77, PH98]. In particular, there exists a generator matrix of $\mathcal{C}$ under the form:

$$G = \begin{pmatrix} g_0 & \cdots & g_m & 0 & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_m & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & g_0 & \cdots & g_m \end{pmatrix} \tag{2}$$

When it is monic, the polynomial $g(X) = g_0 + g_1X + \cdots + g_mX^m$ is called *the generator polynomial of the code $\mathcal{C}$*. This polynomial is a divisor of $X^{2m} - 1$ and there is a vector-space isomorphism between codewords of $\mathcal{C}$ and polynomials of the form $u(X)g(X) \bmod X^{2m} - 1$. Any monic irreducible polynomial g dividing $X^{2m} - 1$ uniquely defines its corresponding cyclic code.

In [AF14] it is additionnally shown that if g is has degree m and generates a cyclic code of length $2m$, then a generator matrix under systematic form of the code is given by:

$$\left(\begin{array}{cc|cccc} X^m & \bmod g & 1 & 0 & \cdots & 0 \\ X^{m+1} & \bmod g & 0 & 1 & \cdots & 0 \\ & \vdots & \vdots & \ddots & \ddots & \vdots \\ X^{2m-1} & \bmod g & 0 & 0 & \cdots & 1 \end{array}\right)$$

We denote by M_g the matrix defined by:

$$M_g = \begin{pmatrix} X^m & \bmod g \\ X^{m+1} & \bmod g \\ & \vdots \\ X^{2m-1} & \bmod g \end{pmatrix} \tag{3}$$

Theorem 1. *Let g be a polynomial of degree m. Then M_g is the m^{th} power of the companion matrix associated with g:*

$$M_g = C_g^m$$

If g is the generator polynomial of a $[2m, m]$-cyclic MDS code, then M_g is both MDS and involutive.

$$M_g^2 = I_m$$

Proof. We denote $GF(2^{2m})[X]/(g(X))$ by E_g. Its canonical basis is $\{1, X, \ldots, X^{m-1}\}$.

The proof of the first part of the theorem is an imbrication of the two following facts:

- M_g is the matrix of the multiplication by X^m in E_g in its canonical basis.
- C_g is the matrix of the multiplication by X in E_g in its canonical basis.

Let now g be the generator polynomial of a $[2m, m]$-cyclic code. By construction, M_g is then the redundancy matrix of a generator matrix under systematic form of an MDS code. By definition, M_g is then MDS.

The proof of the involutivity of M_g is the imbrication of the two following facts:

- M_g is the matrix of multiplication by X^m in E_g in its canonical basis.
- The multiplication by X^m in E_g is an involution since g divides $X^{2m} - 1$: For any polynomial $f(X) \in E_g$, $X^{2m} f(X) \bmod g = f(X) \bmod g$.

□

Remark 1.

1. As a consequence of the theorem, from a $[2m, m]$-cyclic codes we could derive an involutive, recursive, MDS matrix. However, this reveals of little use since most applications are made in characteristic 2. Known direct constructions of cyclic codes are indeed BCH codes and, as was proved in [AF14], it is not possible to find MDS BCH codes of length $2m$ and dimension m since it then needs the existence of a root of even order.
2. Furthermore, for matrix sizes $2m = 2^s$, we know that there is no $[2m, m]$ cyclic MDS code in characteristic 2. Such a code would be generated by a polynomial that divides $X^{2^{2s}} - 1$. However, $X^{2^{2s}} - 1$ factorizes in $(X - 1)^{2^s}$. The only polynomial that generates a $[2m, m]$ cyclic code is then $X^{2^{s-1}} - 1$. This code cannot be MDS since the redundancy matrix has zero coefficients in its first line. Thus there are no cyclic $[8, 4]$ or $[16, 8]$ MDS code, implying that 4×4 or 8×8 MDS matrices of the form (3) cannot be obtained from direct construction via cyclic codes.
3. Augot and Finiasz method deals with this difficulty to derive from this property a direct construction of MDS matrices of the form (3) by considering longer codes they shorten. This method, however, does not preserve involutivity anymore.
4. The relation between M_g and C_g is of great interest for implementation since it can be in hardware efficiently recursively implemented by clocking an LFSR m times.

A conclusion to the section is:

- MDS involutive recursive matrices of order 4×4, 8×8, $16 \times 16, \ldots$ do not exist over fields of characteristic 2. Therefore there is no hope to obtain the direct and inverse product of such a matrix by clocking the same LFSR.

- No direct construction of MDS recursive matrix is known.

In the following, we show that, by skewing slightly the multiplication, we can obtain direct constructions of MDS involutive matrices of some recursive type.

3 2-cyclic codes and 2-polynomial rings

We extend cyclic codes-like properties by *skewing* the notion of cyclicity, preserving the interest of the recursive structure and obtaining a direct construction for MDS matrices. This *skewing* operation consists in authorizing the squaring operations on symbols in the finite field $GF(2^{2m})$. This generalisation of the structure of cyclicity is called 2-cyclicity.

Definition 2 (2-cyclic codes, [Gab85])**.** Let $\mathcal{C}$ be a linear code of length n over $GF(2^{2m})$. Then $\mathcal{C}$ is 2-cyclic if for any vector $\mathbf{x} = (x_0, \ldots, x_{2m-1}) \in \mathcal{C}$, the rotated and squared vector:

$$(\mathbf{x} \ggg 1)^{[1]} = \left(x_{2m-1}^{[1]}, x_0^{[1]}, \ldots, x_{2m-2}^{[1]}\right)$$

belongs to $\mathcal{C}$.

From [Gab85] there exists a generator matrix of $\mathcal{C}$ under the form:

$$G = \begin{pmatrix} g_0 & \cdots & g_m & 0 & 0 & \cdots & 0 \\ 0 & g_0^{[1]} & \cdots & g_m^{[1]} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & g_0^{[m-1]} & \cdots & g_m^{[m-1]} \end{pmatrix} \tag{4}$$

The matrix G is formed with rows such that the $(i+1)$th row is obtained by shifting and squaring the ith row. Of course and contrarily to the classical case of cyclic codes as seen in the previous section, the polynomial $g(X) = g_0 + g_1X + \cdots + g_mX^m$ does not divide $X^{2m} - 1$ in the usual sense, but in a *skewed sense.*

This property is very much related to properties of one specific polynomial ring. This polynomial ring can be found under several denominations and was originally introduced by Ore [Ore33, Ore34] who did not give a particular denomination. In [Ber84], it is called linearized polynomial ring since it comes with an evaluation map which makes the ring isomorphic to the ring of linear mappings over finite fields. More generally, in the field of symbolic computation, it is denoted by skew polynomial ring or Ore ring, [BP94]. What is important to understand is that it has *almost* all the nice properties to perform efficient arithmetic operations except the commutativity property.

Focusing on a ground field with characteristic 2, we call 2-*polynomial ring* and denote by $GF(2^{2m})\langle X\rangle$ the ring defined as the set of polynomials $\sum_i a_iX^i$ together with the following operations:

- *Addition* : classical addition of polynomials.
- *Multiplication by an element in* $GF(2^{2m})$: $X * a = a^{[1]} * X$.

Note that if we remove the $^{[1]}$ in the exponent, we get back to the classical polynomial ring. To differentiate 2-polynomials from classical polynomials, we denote a 2-polynomial A by $A\langle X\rangle$ rather than $A(X)$.

Here, the multiplication is skewed by the squaring. This could be a problem, but it preserves very useful properties for finite field arithmetic. This ring is left and right Euclidean:

Given $A\langle X\rangle$ and $B\langle X\rangle$ in $GF(2^{2m})\langle X\rangle$, there exists unique 2-polynomial pairs (q_1, r_1) and (q_2, r_2) such that:

- $A\langle X\rangle = B\langle X\rangle * q_1\langle X\rangle + r_1\langle X\rangle$, where $\deg(r_1) < \deg(B)$
- $A\langle X\rangle = q_2\langle X\rangle * B\langle X\rangle + r_2\langle X\rangle$, where $\deg(r_2) < \deg(B)$

Example 1. Here a small example to understand how to manipulate this ring. Consider for instance $P\langle X\rangle = X + 1$ and $Q\langle X\rangle = X + \alpha$, where α lies in some field of characteristic 2. We have

- $P\langle X\rangle * Q\langle X\rangle = X^2 + (\alpha^2 + 1)X + \alpha$
- $Q\langle X\rangle * P\langle X\rangle = X^2 + (\alpha + 1)X + \alpha$

Therefore $P\langle X\rangle * Q\langle X\rangle = Q\langle X\rangle * P\langle X\rangle$ if and only if $\alpha^2 = \alpha$, that is $\alpha = 0$ or $\alpha = 1$.

Now we return to the similarities between the 2-cyclic codes and the cyclic codes. If $\mathcal{C}$ is a 2-cyclic code, then it has a generator matrix under the form (4) where the 2-polynomial $g\langle X\rangle = g_0 + g_1X + \cdots + g_mX^m$, $g_m \neq 0$ divides $X^{2m} - 1$ on the right (*i.e.* $X^{2m} - 1 = h\langle X\rangle * g\langle X\rangle$).

As in the classical case, the codewords of $\mathcal{C}$ correspond to the coefficients of the 2-polynomials $u\langle X\rangle * g\langle X\rangle$, where $u\langle X\rangle$ runs through all 2-polynomials of degree $\leq n - m$.

We denote by $\text{mod}_* g$ the operation of computing the remainder of the Euclidean division on the right by g:

$$c\langle X\rangle \ \text{mod}_* g = r\langle X\rangle \Leftrightarrow c\langle X\rangle = b\langle X\rangle * g\langle X\rangle + r\langle X\rangle$$

According to this notation, a generator matrix under systematic form of $\mathcal{C}$ is:

$$\left(\begin{array}{cc|cccc} X^m & \text{mod}_* g & 1 & 0 & \cdots & 0 \\ X^{m+1} & \text{mod}_* g & 0 & 1 & \cdots & 0 \\ \multicolumn{2}{c|}{\vdots} & \vdots & \ddots & \ddots & \vdots \\ X^{n-1} & \text{mod}_* g & 0 & 0 & \cdots & 1 \end{array}\right), \tag{5}$$

We denote by N_g the matrix defined by:

$$N_g = \begin{pmatrix} X^m & \text{mod}_* g \\ X^{m+1} & \text{mod}_* g \\ & \vdots \\ X^{2m-1} & \text{mod}_* g \end{pmatrix} \tag{6}$$

The following theorem proves in the 2-cyclic case, results similar to Theorem 1.

Theorem 2. *Let g be a 2-polynomial of degree m. Then N_g satisfies the* quasi-recursive *property,* i.e. *is the product of the squared powers of companion matrices associated with g:*

$$N_g = C_g^{[m-1]} \cdots C_g^{[1]} C_g = C_{g^{[m-1]}} \cdots C_{g^{[1]}} C_g,$$

where $g^{[i]}\langle X\rangle = \sum_{j=0}^{m-1} g_j^{[i]} X^j + X^m.$

If g is the generator polynomial of a $[2m, m]$ 2-cyclic MDS code, then N_g is MDS and satisfies the quasi-involutory *property:*

$$N_g^{[m]} N_g = I_m$$

Proof. We denote $GF(2^{2m})\langle X\rangle/(g\langle X\rangle)$ by F_g. Here we consider the remainder of the right division by $g\langle X\rangle$. Its canonical basis is $\{1, X, \dots, X^{m-1}\}$.

We will show by induction that for all integers $i \geq 1$, the following property is satisfied:

$$\mathcal{P}_i : \begin{pmatrix} X^i & \bmod_* g \\ & \vdots \\ X^{i+m-1} & \bmod_* g \end{pmatrix} = C_g^{[i-1]} \dots C_g$$

The property $\mathcal{P}_0$ is satisfied. Namely, the matrix is the companion matrix associated with g:

$$\begin{pmatrix} X^1 & \bmod_* g \\ & \vdots \\ X^m & \bmod_* g \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ g_0 & g_1 & \dots & \dots & g_{m-1} \end{pmatrix}$$

Suppose now that $\mathcal{P}_i$ is satisfied for some $i \geq 1$. We have then:

$$\mathcal{P}_i : \begin{pmatrix} X^i & \bmod_* g \\ & \vdots \\ X^{i+m-1} & \bmod_* g \end{pmatrix} = C_g^{[i-1]} \dots C_g$$

Thus, computing $C_g^{[i]} C_g^{[i-1]} \dots C_g$, we first observe that the $(m-1)$ first lines are respectively equal to $X^{i+1} \bmod_* g, \dots, X^{i+m-1} \bmod_* g$.

The last line satisfies:

$$\begin{aligned} \sum_{j=0}^{m-1} g_j^{[i]} (X^{i+j} \bmod_* g) &= (\sum_{j=0}^{m-1} g_j^{[i]} X^{i+j}) \bmod_* g &= \left(X^i (\sum_{j=0}^{m-1} g_j X^j) \right) \bmod_* g \\ &= \big(X^i * (X^m + g(X))\big) \bmod_* g &= X^{i+m} \bmod_* g \end{aligned}$$

which concludes the proof by induction.

From that property, we obtain directly the first part of the theorem.

Let now g be the generator polynomial of a $[2m, m]$ 2-cyclic code.

P_{2m} may be written :

$$C_g^{[2m-1]} \dots C_g = \begin{pmatrix} X^{2m} & \bmod_* g \\ & \vdots \\ X^{3m-1} & \bmod_* g \end{pmatrix}$$

As $C_g^{[2m-1]} \dots C_g = C_g^{[2m-1]} \dots C_g^{[m]} C_g^{[m-1]} \dots C_g = (C_g^{[m-1]} \dots C_g)^{[m]} (C_g^{[m-1]} \dots C_g)$, we have :

$$N_g^{[m]} N_g = \begin{pmatrix} X^{2m} & \bmod_* g \\ & \vdots \\ X^{3m-1} & \bmod_* g \end{pmatrix}$$

Recall now that g divides on the right $X^{2m} - 1$. Then, for any polynomial $f(X) \in F_g$, $f(X) X^{2m} \bmod_* g = f(X) \bmod_* g$. We obtain then:

$$N_g^{[m]} N_g = \begin{pmatrix} X^0 & \bmod_* g \\ & \vdots \\ X^{m-1} & \bmod_* g \end{pmatrix} = I_m$$

Finally, N_g is by construction the redundancy matrix of a generator matrix under systematic form of an MDS code. By definition, N_g is then MDS.

□

4 Direct construction

This section exploits the results of Theorem 2 to directly construct a new family of MDS matrices which has the property to be *quasi-involutory*. In a first part, from Gabidulin's results, we show how to directly construct MDS matrices under the form (6), satisfying the *quasi-involutory* property. Finally we propose an algorithmic procedure to generate them, and show that the matrices generated with this procedure are all different.

4.1 2-cyclic Gabidulin codes

The 2-cyclic MDS Gabidulin codes are constructed as follows, [Gab85]: Consider the finite field $GF(2^{2m})$ (typically in the following $2m = 8$). Let α be a normal element of the field. This means that $\mathcal{B} = \{\alpha, \alpha^{[1]}, \ldots, \alpha^{[2m-1]}\}$ is a binary basis of $GF(2^{2m})$. Then the code $\mathcal{C}$ having the following parity-check matrix is MRD and therefore MDS:

$$H = \begin{pmatrix} \alpha^{[0]} & \alpha^{[1]} & \cdots & \alpha^{[2m-1]} \\ \alpha^{[1]} & \alpha^{[2]} & \cdots & \alpha^{[0]} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{[m-1]} & \alpha^{[m]} & \cdots & \alpha^{[m-2]} \end{pmatrix} \tag{7}$$

Additionally, Gabidulin proved that the code $\mathcal{C}$ has a generator-matrix

$$G = \begin{pmatrix} g_0 & \cdots & g_m & 0 & 0 & \cdots & 0 \\ 0 & g_0^{[1]} & \cdots & g_m^{[1]} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & g_0^{[m-1]} & \cdots & g_m^{[m-1]} \end{pmatrix}$$

where the elements $g_i \in GF(2^{2m})$ satisfy the following linear system:

$$\sum_{i=0}^{m} g_i \alpha^{[i+j]} = 0, \ \forall j = 0, \ldots, m-1. \tag{8}$$

what implies that the 2-polynomial $g\langle X\rangle = \sum_{i=0}^{m} g_i X^i$, is a right divisor of $X^{2m} - 1$. Therefore the code $\mathcal{C}$ is 2-cyclic, MDS and generated by g. Therefore,

$$\mathcal{C} = \langle G \rangle = \langle (N_g \mid I_m) \rangle,$$

And by using Theorem 2 we conclude that the matrix:

$$N_g = \begin{pmatrix} X^m & \mod_* g \\ X^{m+1} & \mod_* g \\ & \vdots \\ X^{2m-1} & \mod_* g \end{pmatrix} \tag{9}$$

is MDS, satisfying the *quasi-involutory* and *quasi-recursive* properties.

4.2 General construction

We now show how to construct such MDS matrices N_g from 2-cyclic MDS Gabidulin codes by considering normal elements:

- Choose an element $\alpha \in GF(2^{2m})$ and check if it is normal (different algorithms exist to test the normality of an element). Since we usually consider m as a power of 2, $2m$ is also a power of two and this implies that there are exactly 2^{2m-1} such normal elements, which are exactly the elements of trace equal to 1, [MP13] corollary 5.2.9. Therefore, half of the elements of the field are normal. If $2m$ is not a power of 2, then the number of normal elements is known, [MP13] corollary 5.2.8.

- Generate the matrix $H = (H_1 \mid H_2)$ such that:

$$H_1 = \begin{pmatrix} \alpha^{[0]} & \alpha^{[1]} & \cdots & \alpha^{[m-1]} \\ \alpha^{[1]} & \alpha^{[2]} & \cdots & \alpha^{[m]} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{[m-1]} & \alpha^{[m]} & \cdots & \alpha^{[2m-1]} \end{pmatrix}, \; H_2 = \begin{pmatrix} \alpha^{[m]} & \alpha^{[m+1]} & \cdots & \alpha^{[2m-1]} \\ \alpha^{[m+1]} & \alpha^{[m+2]} & \cdots & \alpha^{[0]} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{[2m-1]} & \alpha^{[0]} & \cdots & \alpha^{[m-2]} \end{pmatrix}.$$

- Set $N_g = H_2 H_1^{-1}$.

To be done, it suffices to prove that the matrix $H = (H_1, \mid H_2)$ is a parity-check matrix of the code $\mathcal{C}$ generated by $(H_2 H_1^{-1} \mid I_m)$. We have:

$$(H_2 H_1^{-1} \mid I_m)(H_1 \mid H_2)^T = H_2 H_1^{-1} H_1^T + H_2^T.$$

Now since H_1 and H_2 are symmetric, $H_1^T = H_1$ and $H_2^T = H_2$. Therefore, we have

$$H_2 H_1^{-1} H_1^T + H_2^T = H_2 + H_2 = 0,$$

in characteristic 2.

Remark 2. Once such a matrix is constructed, any matrix of the form $N_g^{[i]}$ is also *quasi-involutory*. Another nice property relies on the fact that if C is a circulant non-singular binary matrix then CM is an MDS matrix that satisfies $(CM)^{[m]}CM = C^2$, therefore the product is not anymore involutive but its coefficients belong to the binary field.

The following theorem shows that all constructed matrices are different, if we depart from different normal elements.

Theorem 3. *Let α_1 and α_2 two distinct normal elements, then the matrices N_g obtained by the previous construction are different.*

Proof. Let $\mathcal{C}_{\alpha_1}$ and $\mathcal{C}_{\alpha_2}$, the codes with parity check matrices $H = (H_1 \mid H_2)$ respectively constructed from α_1 and α_2. The respective MDS matrices say $N_g(\alpha_1)$ and $N_g(\alpha_2)$ are equal if and only if $\mathcal{C}_{\alpha_1} = \mathcal{C}_{\alpha_2}$, that is if and only if the generating polynomials $g_{\alpha_1}\langle X\rangle$ and $g_{\alpha_2}\langle X\rangle$ are equal. The reason is that a 2-cyclic code is uniquely defined by a monic generating polynomial.

From a linear application point of view [Ber84], the root space of g_{α_1} is $\langle \alpha_1, \dots, \alpha_1^{[m-1]}\rangle$ and the root space of g_{α_2} is $\langle \alpha_2 \dots, \alpha_2^{[m-1]}\rangle$. Therefore, $N_g(\alpha_1) = N_g(\alpha_2)$ implies

$$\mathcal{V} = \langle \alpha_1, \dots, \alpha_1^{[m-1]}\rangle = \langle \alpha_2, \dots, \alpha_2^{[m-1]}\rangle.$$

Suppose now that this equality is satisfied. Then α_1 is a binary linear combination of $\alpha_2, \dots, \alpha_2^{[m-1]}$:

$$\alpha_1 = \sum_{i=0}^{m-1} a_i \alpha_2^{[i]}, \; a_i \in GF(2).$$

Since the squaring operation is 2-linear in fields of characteristic 2, we obtain

$$\alpha_1^{[1]} = \sum_{i=0}^{m-1} a_i\alpha_2^{[i+1]} = \underbrace{\sum_{i=1}^{m-1} a_{i-1}\alpha_2^{[i]}}_{\in\mathcal{V}} + a_{m-1}\alpha_2^{[m]}.$$

Now since the first term $\in \mathcal{V}$, it can be rewritten as a binary linear combination of $\alpha_1, \dots, \alpha_1^{[m-1]}$. Considering the second term $\alpha_2^{[m]}$, it can be rewritten as $(\alpha_2^{[m-1]})^{[1]}$, where $\alpha_2^{[m-1]} = \sum_{i=0}^{m-1} b_i\alpha_1^{[i]} \in \mathcal{V}$. Therefore $\alpha_2^{[m]} = \sum_{i=0}^{m-1} b_{i-1}\alpha_1^{[i]} + b_{m-1}\alpha_1^{[m]}$. Now necessarily $b_{m-1} = 1$. Namely, if $b_{m-1} = 0$, this would imply that $\alpha_2^{[m]} \in \mathcal{V}$, but since α_2 is normal by hypothesis, $\alpha_2^{[m]}$ cannot be a binary linear combination of other squared powers of α_2. Therefore, we have

$$\alpha_1^{[1]} = \underbrace{\lambda}_{\in\mathcal{V}} + a_{m-1}\alpha_1^{[m]}.$$

The term λ is a binary linear combination of $\alpha_1, \dots, \alpha_1^{[m-1]}$. Since α_1 is normal, then $\alpha_1, \dots, \alpha_1^{[m-1]}, \dots, \alpha_1^{[2m-1]}$ is a binary basis of $GF(2^{2m})$. Therefore, $a_{m-1} = 0$. By induction, we show that $a_i = 0$ for $i \neq 0$. Therefore, if $\langle \alpha_1, \dots, \alpha_1^{[m-1]}\rangle = \langle \alpha_2, \dots, \alpha_2^{[m-1]}\rangle$, then necessarily $\alpha_1 = \alpha_2$.

Hence if $\alpha_1 \neq \alpha_2$, then $N_g(\alpha_1) \neq N_g(\alpha_2)$.

□

Remark 3. In the proof we omitted some arguments explaining why the root space uniquely determines the polynomial. The point is that g_{α_1} and g_{α_2} have 2-degree m and annihilate binary vector spaces of dimension m. Since they are monic, the theory of 2-polynomials establishes the unicity, as in the classical case where, given m distinct points in some field, there is a unique monic polyomial of degree m whose roots are the m distinct points.

We sum up here the routine to generate an MDS matrix according to our construction :

1. Choose a normal element α.
2. Compute $H = (H_1 \mid H_2)$ with:

$$H_1 = \begin{pmatrix} \alpha^{[0]} & \alpha^{[1]} & \cdots & \alpha^{[m-1]} \\ \alpha^{[1]} & \alpha^{[2]} & \cdots & \alpha^{[m]} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{[m-1]} & \alpha^{[m]} & \cdots & \alpha^{[2m-1]} \end{pmatrix},\ H_2 = \begin{pmatrix} \alpha^{[m]} & \alpha^{[m+1]} & \cdots & \alpha^{[2m-1]} \\ \alpha^{[m+1]} & \alpha^{[m+2]} & \cdots & \alpha^{[0]} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{[2m-1]} & \alpha^{[0]} & \cdots & \alpha^{[m-2]} \end{pmatrix}.$$

3. Compute $N_g = H_2H_1^{-1}$. The inverse matrix is $N_g^{-1} = N_g^{[m]}$.
4. Compute C_g from the first line of N_g.

Now we conclude the section with an example.

Example 2. Let $m = 4$. The finite field we consider is then $GF(2^8)$. Let β be a generator of the multiplication group $GF(2^8) = \{1, \beta, \dots, \beta^{254}\}$ where β is a root of the irreducible polynomial $x^8 + x^4 + x^3 + x^2 + 1 = \texttt{0x11c}$. Note that β is not a normal element.

1. We chose to consider the normal element β^{21}.
2. The parity-check matrix under the form (7) is:

$$\left(\begin{array}{c|c|c|c|c|c|c|c} \beta^{21} & \beta^{42} & \beta^{84} & \beta^{168} & \beta^{81} & \beta^{162} & \beta^{69} & \beta^{138} \\ \beta^{42} & \beta^{84} & \beta^{168} & \beta^{81} & \beta^{162} & \beta^{69} & \beta^{138} & \beta^{21} \\ \beta^{84} & \beta^{168} & \beta^{81} & \beta^{162} & \beta^{69} & \beta^{138} & \beta^{21} & \beta^{42} \\ \beta^{168} & \beta^{81} & \beta^{162} & \beta^{69} & \beta^{138} & \beta^{21} & \beta^{42} & \beta^{84} \end{array}\right)$$

3. Hence the MDS matrix N_g is written:

$$N_g = \left(\begin{array}{c|c|c|c} \beta^{199} & \beta^{96} & \beta^{52} & \beta^{123} \\ \beta^{190} & \beta^{218} & \beta^{231} & \beta^{125} \\ \beta^{194} & \beta^{227} & \beta^{224} & \beta^{66} \\ \beta^{76} & \beta^{54} & \beta^{217} & \beta^{28} \end{array}\right) = \left(\begin{array}{c|c|c|c} \texttt{0x0e} & \texttt{0xd9} & \texttt{0x14} & \texttt{0xc5} \\ \texttt{0xae} & \texttt{0x2b} & \texttt{0xf5} & \texttt{0x33} \\ \texttt{0x32} & \texttt{0x90} & \texttt{0x12} & \texttt{0x61} \\ \texttt{0x1e} & \texttt{0x50} & \texttt{0x9b} & \texttt{0x18} \end{array}\right)$$

The second matrix being the expression of N_g in hexadecimal in the basis `0x11c`

Its inverse matrix is:

$$N_g^{-1} = N_g^{[4]} = \left(\begin{array}{c|c|c|c} \beta^{124} & \beta^{6} & \beta^{67} & \beta^{183} \\ \beta^{235} & \beta^{173} & \beta^{126} & \beta^{215} \\ \beta^{44} & \beta^{62} & \beta^{14} & \beta^{36} \\ \beta^{196} & \beta^{99} & \beta^{157} & \beta^{193} \end{array}\right) = \left(\begin{array}{c|c|c|c} \texttt{0x97} & \texttt{0x40} & \texttt{0xc2} & \texttt{0xc4} \\ \texttt{0xeb} & \texttt{0xf6} & \texttt{0x66} & \texttt{0xef} \\ \texttt{0xee} & \texttt{0xde} & \texttt{0x13} & \texttt{0x25} \\ \texttt{0xc8} & \texttt{0x96} & \texttt{0xd5} & \texttt{0x19} \end{array}\right)$$

4. As we will see in Section 5, the matrix-vector product with N_g may be done by implementing only the companion matrix associated with g, the unique monic 2-polynomial of degree 4 satisying the linear equations, $g\langle X\rangle = \beta^{199} + \beta^{96}X + \beta^{52}X^2 + \beta^{123}X^3 + X^4$:

$$C_g = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \beta^{199} & \beta^{96} & \beta^{52} & \beta^{123} \end{pmatrix} = \left(\begin{array}{c|c|c|c} \texttt{0x00} & \texttt{0x01} & \texttt{0x00} & \texttt{0x00} \\ \texttt{0x00} & \texttt{0x00} & \texttt{0x01} & \texttt{0x00} \\ \texttt{0x00} & \texttt{0x00} & \texttt{0x00} & \texttt{0x01} \\ \texttt{0x0e} & \texttt{0xd9} & \texttt{0x14} & \texttt{0xc5} \end{array}\right)$$

5 Implementation

The goal of this section is to show that the direct product by a matrix N_g under the form (6) can be implemented by clocking a *skewed* LFSR. Whenever the matrix is *quasi-involutory* the inverse product is implemented by adding some routing.

First we present two simple algorithms computing the direct and inverse product of a vector by a matrix N_g under the form (6). The core of the algorithms comes from the decomposition presented in Theorem 2 and consists in iterating the same loop. A step in the loop corresponds to some sort of product with a companion matrix of a polynomial. In a second part we show that this product can be efficiently implemented by clocking a *skewed* LFSR. Then we present the results we obtained in hardware implementation. In a final part we make some comparisons of our construction with some other existing constructions by using the metric derived from the so-called XOR count. With our direct constructions of MDS matrices from MDS Gabidulin codes, we did not find matrices that have better XOR counts than best known constructions. It is not a surprise since direct constructions are a lot more restrictive than exhaustive search. Our goal is more to give an alternative to previous constructions of MDS matrices than to try to improve the best results.

However to validate the interest of the architecture in itself, for small parameters we did an exhaustive search on matrices constructed as product of powers of companion matrices and computed the XOR count involved by clocking the register. With this method we were able to show that this construction was competitive with the best constructions found in the literature.

5.1 Algorithm

Algorithms 1 and 2 compute matrix-vector product respectively for a matrix N_g and its inverse where the matrix N_g is of the form :

$$N_g = C_g^{[m-1]} \cdots C_g$$

Algorithm 1 Matrix vector product

```
Require: $\mathbf{x} \in GF(2^{2m})^m$ an input vector and $C_g$
Ensure: $\mathbf{y} = N_g\mathbf{x}$ the result of the matrix vector product
1: $\mathbf{y} \leftarrow \mathbf{x}^{[1]}$                                   ▷ Initialization
2: for $i = 0$ to $m - 1$ do
3:     $\mathbf{y} \leftarrow C_g\mathbf{y}^{[-1]}$          ▷ Matrix-vector product with companion matrix
4: end for
5: $\mathbf{y} \leftarrow \mathbf{y}^{[m-1]}$                                 ▷ Final step
6: return $\mathbf{y}$
```

The proof of correctness can be made by induction. We show that after the ith step of the *for* loop, the following property is satisfied :

$$\mathcal{P}_i : \ \mathbf{y}^{[i]} = C_g^{[i]} \cdots C_g\mathbf{x}$$

The property $\mathcal{P}_0$ is satisfied. Namely, we have $\mathbf{y} = C_g\mathbf{x}$.

Suppose now that $\mathcal{P}_i$ is satisfied for some i. After the $(i+1)$th step, we have then, thanks to the distributive property of the squaring operator :

$$\mathbf{y}_{(i+1)}^{[i+1]} = \left(C_g\mathbf{y}_{(i)}^{[-1]}\right)^{[i+1]} = C_g^{[i+1]}\mathbf{y}_{(i)}^{[i]} = C_g^{[i+1]}C_g^{[i]} \cdots C_g\mathbf{x}$$

Therefore, $\mathcal{P}_{i+1}$ is also satisfied.

By induction, we have then after m step :

$$\mathcal{P}_{m-1} : \ \mathbf{y}^{[m-1]} = C_g^{[m-1]} \cdots C_g\mathbf{x}$$

Therefore, at the end of the loop, we have to square the result $m-1$ times to get the wanted matrix-vector product, explaining thus the *Final step*.

Algorithm 2 Matrix-vector product for the inverse matrix

```
Require: $\mathbf{x} \in GF(2^{2m})^m$ an input vector and $C_g$
Ensure: $\mathbf{y} = N_g^{-1}\mathbf{x}$ the result of the matrix vector product
1: $\mathbf{y} \leftarrow \mathbf{x}^{[-m+1]}$                                ▷ Initialization
2: for $i = 0$ to $m - 1$ do
3:     $\mathbf{y} \leftarrow C_g\mathbf{y}^{[-1]}$          ▷ Matrix-vector product with companion matrix
4: end for
5: $\mathbf{y} \leftarrow \mathbf{y}^{[-1]}$                                  ▷ Final step
6: return $\mathbf{y}$
```

The proof of correctness is almost the same since :

$$N_g^{-1}\mathbf{x} = N_g^{[m]}\mathbf{x} = \left(N_g\mathbf{x}^{[-m]}\right)^{[m]}$$

Thus, computing the matrix-vector product and the matrix-vector product with the inverse matrix can be factorized easily. Only initialization and final steps are different but we now show that it can be done very efficiently in hardware.

5.2 Hardware Architecture

In the previous sections, we demonstrated how to directly construct MDS matrices with *quasi-involutory* property. The direct and inverse product computation can be done by clocking an SLFSR. In this section we show that this SLFSR architecture is interesting in hardware implementations compared to the direct matrix-vector product when considering growing size matrices.

The specific form of the matrix N_g can be computed using the architecture presented in Fig. 1. This architecture is derived from LFSR architecture. It is composed of registers (X_i), $\gamma^{[-1]}$ units, multipliers in Galois fields by constants (g_i) and the sum in Galois fields $(xor\text{s})$.

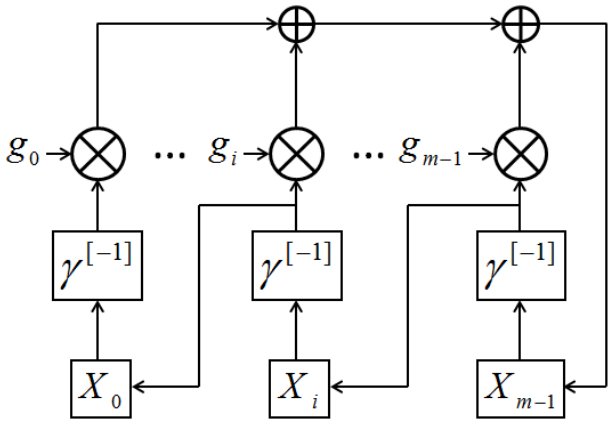

Figure 1: Skewed LFSR Architecture.

The $\gamma^{[-1]}$ units compute the inverse square of X_i. Using an appropriate basis of X_i, a normal basis, the Frobenius map can be evaluated easily using a fixed bit permutation. Thus the $\gamma^{[-1]}$ computation has no cost in hardware resources.

In the design of a cryptographic scheme, a designer would naturally do all the computations in a normal basis which is much more adapted to the design of skewed-LFSRs rather than polynomial bases and there would be no additional cost. As noticed in [CR15], the choice of the basis has "no influence on the differential and linear properties of the cipher", on the complexity of statistical attacks for Substitution-Permutation Networks. Any supplementary cost concerning basis transformations is then implementation dependent and cannot be evaluated a priori. However, if necessary, the transformation from and to a normal basis, linear matrix with fixed coefficient in $GF(2)$, can be done once before and after the execution of the algorithm.

Suppose that $\mathbf{y} = (y_0, \ldots, y_{m-1})$ is the current state of this architecture, then the step obtained by clocking one time the SLFSR is $C_g\mathbf{y}^{[-1]}$. This corresponds to one step of the *for* loop in Algorithms 1 and 2. It implies that the product of matrix N_g by a vector can be simply implemented by clocking m times a structure very close to an LFSR.

Remark 4. The cost of the computation of the matrix-vector product with N_g and with its inverse are the same. However, to be able to compute both with such implementation, one additional MUX is needed to decide which permutation of the input and which permutation of the output are to be done.

5.3 Hardware Results

This section aims to present the validity of our concept. One future work could be to go through exhaustive search to determine if new MDS matrices with lower costs than the

Table 1: P&R performances and comparisons

Design	m	Slices	LUT	FF	Cycles
Unrolled	4	46	138	32	1
	8	483	1657	128	1
SLFSR	4	55	129	46	4
	8	318	826	156	8

state-of-the-art may be found with these direct constructions. These research may reveal costly.

Target technology The design was implemented on a Xilinx Kintex-7 FPGA using the SAKURA evaluation board available. This board includes the device xc7k160t which is a mid range FPGA on the 28nm process node.

Implementation We have validated our approach, using VHDL, by computing the matrix-vector product, with coefficients in $GF(2^{2m})$ with $m = 4$ and $m = 8$, in 2 differents ways:

- unrolled matrix-vector product,
- SLFSR.

Table 1 shows P&R results of both designs. In the case where $m = 4$, one can see that the unrolled product is smaller, in slices, than the SLFSR. This is not surprising because of the relative cost of multiplexers, higher in smaller dimension with SLFSR design. But considering higher dimension, we observe that this cost is compensated by the compact form of the algorithm.

5.4 XOR count and comparisons with previous constructions

In this section, we do not consider matrices from our direct constructions anymore but we present results found through exhaustive searches on all 3×3, 4×4 and 6×6 matrices over $GF(2^4)$ constructed as the product of square powers of companion matrices. For every such matrix, we tested if it was MDS and if it was both MDS and *quasi-involutory*. We computed the *XOR count* of the product of the elements of the register in all the different $2^3 = 8$ normal bases. By considering that all computations outside the computation of the matrix-vector product are made in a normal basis, we do not need to add an extra cost corresponding to a basis input/output transformation. In our simulations, $GF(2^4)$ is generated by the polynomial $x^4 + x + 1 =$ `0x13`.

3×3 matrices

We consider $N_g = C_g^{[2]} C_g^{[1]} C_g$ matrices build from companion matrices :

$$C_g = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ g_0 & g_1 & g_2 \end{pmatrix}$$

We obtained 2010 MDS matrices among which 6 are quasi-involutory. We checked among the list of 1980 MDS matrices obtained by clocking 3 times an LFSR that none of our matrices can be obtained in the classical way.

The companion matrix

$$C_{g_1} = \begin{pmatrix} \texttt{0x0} & \texttt{0x1} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x1} \\ \texttt{0x6} & \texttt{0x6} & \texttt{0x1} \end{pmatrix}$$

generates an MDS quasi-involutory matrix N_{g_1} with XOR count $12 + 2 \times 4$ for the register, when the chosen normal basis is generated by the element `0x9`.

When the condition of *quasi-involutory* is relaxed, which means that we do not have to compute the inverse matrix anymore, the companion matrix:

$$C_{g_2} = \begin{pmatrix} \texttt{0x0} & \texttt{0x1} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x1} \\ \texttt{0x1} & \texttt{0x8} & \texttt{0x1} \end{pmatrix}$$

generates an MDS matrix N_g. The XOR count of the coefficients of the register are $3 + 2 \times 4$, still in the normal basis generated by `0x9`.

4×4 matrices

We consider $N_g = C_g^{[3]} C_g^{[2]} C_g^{[1]} C_g$ matrices build from companion matrices :

$$C_g = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ g_0 & g_1 & g_2 & g_3 \end{pmatrix}$$

We obtained 3120 MDS matrices among which 240 are quasi-involutory. We checked among the list of 3660 MDS matrices obtained by clocking 4 times an LFSR that none of our matrices can be obtained in the classical way.

The companion matrix

$$C_{g_3} = \begin{pmatrix} \texttt{0x0} & \texttt{0x1} & \texttt{0x0} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x1} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x0} & \texttt{0x1} \\ \texttt{0xd} & \texttt{0x1} & \texttt{0xe} & \texttt{0xb} \end{pmatrix}$$

generates an MDS quasi-involutory matrix N_{g_2} with XOR count $13 + 3 \times 4$ for the register, when the chosen normal basis is generated by the element `0x9`.

When the condition of *quasi-involutory* is relaxed, the companion matrix:

$$C_{g_4} = \begin{pmatrix} \texttt{0x0} & \texttt{0x1} & \texttt{0x0} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x1} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x0} & \texttt{0x1} \\ \texttt{0xf} & \texttt{0x1} & \texttt{0x1} & \texttt{0x8} \end{pmatrix}$$

generates an MDS matrix N_g. The XOR count of the coefficients of the register are $6 + 3 \times 4 = 18$, still in the normal basis generated by `0x9`.

6×6 matrices

We consider $N_g = C_g^{[5]} C_g^{[4]} C_g^{[3]} C_g^{[2]} C_g^{[1]} C_g$ matrices build from companion matrices :

$$C_g = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ g_0 & g_1 & g_2 & g_3 & g_4 & g_5 \end{pmatrix}$$

We obtained 60 MDS matrices. We noticed that all of them were quasi-involutory. We checked among the list of 180 MDS matrices obtained by clocking 6 times an LFSR that none of our matrices could be obtained in the classical way. Among them, the companion matrix:

$$C_{g_6} = \begin{pmatrix} \texttt{0x0} & \texttt{0x1} & \texttt{0x0} & \texttt{0x0} & \texttt{0x0} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x1} & \texttt{0x0} & \texttt{0x0} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x0} & \texttt{0x1} & \texttt{0x0} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x0} & \texttt{0x0} & \texttt{0x1} & \texttt{0x0} \\ \texttt{0x0} & \texttt{0x0} & \texttt{0x0} & \texttt{0x0} & \texttt{0x0} & \texttt{0x1} \\ \texttt{0xa} & \texttt{0x5} & \texttt{0x1} & \texttt{0xa} & \texttt{0xb} & \texttt{0x1} \end{pmatrix}$$

generates an MDS quasi-involutory matrix N_{g_1} with XOR count $17{+}5{\times}4$ for the register, when the chosen normal basis is generated by the element `0x9`. In [LS16], generalising the notion of circulant matrices does not allow the authors to construct MDS involutive matrices. We then see that an SLFSR construction may offer interesting alternatives for certain sets of parameters where involutive MDS matrices are difficult to build.

As shown in Table 2, the obtained results are very similar to the best obtained in the literature. Recall here that for Quasi-Involutory Skewed Recursive matrices, to allow both computations of matrix-vector product with the matrix and with its inverse, the cost of a MUX operator must be added to the XOR count given in this table, while in the case of involutory matrices, the XOR count corresponds to the overall cost.

Table 2: Best known MDS matrices with $\mathbb{F}_{2^4}$ elements without counting the necessary cost of a MUX for quasi-involutory matrices

matrix type	Matrix Size	Ground Field	XOR Count	Reference
Circulant	3×3	$GF(2^4)$	$1+2\times4$	[LS16]
Skewed Recursive	3×3	$GF(2^4)$	$3+2\times4$	this paper
Involutory Circulant	3×3	$GF(2^4)$	$12+2\times4$	[LS16]
Quasi-Involutory Skewed Recursive	3×3	$GF(2^4)$	$12+2\times4$	this paper
Circulant	4×4	$GL(4,GF(2))$	$3+3\times4$	[LW16]
Circulant	4×4	$GF(2^4)$	$3+3\times4$	[LS16]
Skewed Recursive	4×4	$GF(2^4)$	$6+3\times4$	this paper
Involutory Circulant	4×4	$GL(4,GF(2))$	$5+3\times4$	[LW16]
Involutory Hadamard	4×4	$GF(2^{2m})$/`0x13`	$6+3\times4$	[SKOP15]
Quasi-Involutory Skewed Recursive	4×4	$GF(2^4)$	$13+3\times4$	this paper
Circulant	6×6	$GF(2^4)$	$12+5\times4$	[LS16]
Quasi-Involutory Skewed Recursive	6×6	$GF(2^4)$	$17+5\times4$	this paper

All these constructed matrices cannot be obtained by our direct construction, rather by exhaustive search and testing. However, the obtained results are close to the best existing constructions. This demonstrates the interest of constructing MDS matrices by clocking an SLFSR.

Moreover, our method ensures that even in situations were the exhaustive search is not feasible we can construct matrices that are both MDS and *quasi-involutory*.

6 Conclusion and perspectives

In this paper, we give a new direct construction of MDS matrices that may be very efficiently implemented by clocking a so-called SLFSR. This new method relies on non-commutative polynomials where multiplication has been skewed by the squaring operator. Our work may be seen as a generalization of cyclic codes and classical polynomials in the sense that if we choose the identity automorphism to construct our polynomial ring, we get back to the classical theory. As a direct construction, it opens the gate for building big diffusion layers without the costs of exhaustive search. We presented here only the most intuitive construction but we can do exactly the same with any automorphism of the finite field and not only with the squaring operator. We can thus obtain more MDS matrices that benefit from the same advantages concerning implementation.

References

[AF14] D. Augot and M. Finiasz. 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*, pages 3–17, 2014.

[Aid86] A. K. Aidinyan. On matrices with nondegenerate square submatrices. *Problems of Information Transmission*, 22(4):106–108, 1986.

[Ber84] E. R. Berlekamp. *Algebraic Coding Theory.* Aegean Press Part, 1984.

[Ber13] T. P. Berger. Construction of recursive MDS diffusion layers from Gabidulin codes. In *Progress in Cryptology-INDOCRYPT 2013*, volume LNCS 8250, pages 274–285. Springer, 2013.

[BGU07] D. Boucher, W. Geiselmann, and F. Ulmer. Skew cyclic codes. In *Applied Algebra in Engineering, Communication and Computing*, volume 18, pages 379–389, 2007.

[BP94] M. Bronstein and M. Petkovsek. On Ore rings, linear operators and factorisation. *Programming and computer software*, (20):14–18, 1994.

[BU09] D. Boucher and F. Ulmer. Coding with skew polynomial rings. In *Journal of Symbolic Computation*, volume 44, pages 1644–1656, 2009.

[CLU09] L. Chaussade, P. Loidreau, and F. Ulmer. Skew codes of prescribed distance or rank. In *Designs, Codes and Cryptography*, volume 50, pages 267–284, 2009.

[CR15] A. Canteaut and J. Roué. On the behaviors of affine equivalent sboxes regarding differential and linear attacks. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part I*, pages 45–74, 2015.

[DR02] J. Daemen and V. Rijmen. *The Design of Rijndael - AES – The Advanced Encryption Standard.* Springer-Verlag, 2002.

[Gab85] E. M. Gabidulin. Theory of codes with maximal rank distance. In *Problems of Information Transmission*, 1985.

[GPP11] J. Guo, T. Peyrin, and A. Poschmann. The PHOTON family of lightweight hash functions. In *Advances in Cryptology - CRYPTO 2011*, pages 222–239, 2011.

[GPPR11] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. The LED block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 326–341, 2011.

[GR14] K. C. Gupta and I. G. Ray. On constructions of circulant MDS matrices for lightweight cryptography. In *ISPEC 2014*, pages 564–576, 2014.

[LS16] M. Liu and S. M. Sim. Lightweight MDS generalized circulant matrices. In *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 101–120, 2016.

[LW16] Y. Li and M. Wang. 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*, pages 121–139, 2016.

[Men93] A. J. Menezes, editor. *Applications of Finite Fields.* Kluwer Academic Publishers, 1993.

[MP13] G. L. Mullen and D. Panario. *Handbook of Finite Fields.* Chapman & Hall/CRC, 1st edition, 2013.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes.* North-Holland mathematical library. North-Holland Pub. Co. New York, Amsterdam, New York, 1977. Includes index.

[Ore33] Ø. Ore. Theory of non commutative polynomials. *Annals of Mathematics*, pages 480–508, 1933.

[Ore34] Ø. Ore. Contribution to the theory of finite fields. *Transactions of the American Mathematical Society*, 36:243–274, 1934.

[PH98] V. S. Pless and W. C. Huffman, editors. *Handbook of Coding Theory, VOLUME I.* North-Holland, 1998.

[RL89] R. M. Roth and A. Lempel. On MDS codes via Cauchy matrices. In *IEEE transactions on information theory*, volume 35, pages 1314–1319, 1989.

[RS85] R. M. Roth and G. Seroussi. On generator matrices of MDS codes. In *IEEE transactions on information theory*, volume 31, pages 826–830, 1985.

[SKOP15] S. M. Sim, K. Khoo, F. Oggier, and T. Peyrin. Lightweight MDS involution matrices. In *Fast Software Encryption - 21st International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, pages 471–493, 2015.