A New Approach for the Implementation of Binary Matrices Using SLP Applications

Mahdi Sajadieh[1] Mohsen Mousavi[2*]

[1] Dep. of Electrical Engineering, Islamic Azad University (Khorasgan), Isfahan, Iran.

[2] Dep. of Mathematics, Malek Ashtar University of Technology, Isfahan, Iran

Abstract

In this paper, we propose a method for implementing binary matrices with low-cost XOR. First, using a random-iterative method, we obtain a list **S** from a binary matrix **A**. Then, based on the list **S**, we construct a binary matrix **B**. Next, we find a relation between the implementations of **A** and **B**. In other words, using the implementation of the matrix **B**, we get a low-cost implementation for the matrix **A**. Also, we show that the implementation of an MDS matrix **M** is associated with the form of the binary matrix used to construct the binary form of **M**. In addition, we propose a heuristics algorithm to implement MDS matrices. The best result of this paper is the implementation of a 8×8 involutory MDS matrix over 8-bit words with 408 XOR gates. The Paar algorithm is used as an SLP application to obtain implementations of this paper.

Keywords: Shortest Straight-Line Program, MDS **matrix, Implementation**.

1 Introduction

One of the methods for the implementation of binary matrices is based on the shorter linear straight-line programs such as Paar [1] and BP algorithms [2]. Some optimizations over heuristics for short linear programs are given in [3] and [4]. The new idea in [3] is based on the application of permutation matrices and the proposed approach in [4] is an improvement in the selection phases in heuristics for SLP.

Suppose that we need a circuit that computes the system of equations. In this paper, we propose to modify the circuit in such a way the implementation cost of the modified circuit is less than that of the first circuit. In other words, we suggest changing the underlying linear system to achieve a low cost implementation.

E-mail address:
1-m.sajadieh@khuisf.ac.ir ,
2-m.mousavi@mut-es.ac.ir and mohsen_mousavi_235@yahoo.com (*Corresponding Author).

First, we illustrate our contribution with an example. Let we need to compute the following equations where $\mathbf{x} = [x_1, x_2, \cdots, x_8]$ and $\mathbf{y} = [y_1, y_2, \cdots, y_8]$.

$$
\begin{aligned}
y_1 &= x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8\\
y_2 &= x_2 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8\\
y_3 &= x_3 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8\\
y_4 &= x_4 \oplus x_6 \oplus x_7 \oplus x_8\\
y_5 &= x_5 \oplus x_7 \oplus x_8\\
y_6 &= x_6 \oplus x_8\\
y_7 &= x_1 \oplus x_7\\
y_8 &= x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8
\end{aligned}
\qquad
\mathbf{y}^T = \mathbf{x} \cdot \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1\\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1\\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1\\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 1\\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1\\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1\\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1
\end{pmatrix} \tag{1}
$$

Using the Paar algorithm, we get the next implementation for the binary matrix, given in (1), such that the cost of implementation is 12 XOR.

$$
\begin{array}{llll}
1)\, y_6 = x_6 \oplus x_8, & 2)\, y_7 = x_1 \oplus x_7, & 3)\, t_1 = x_7 \oplus x_8, & 4)\, y_5 = t_1 \oplus x_5,\\
5)\, t_2 = t_1 \oplus x_6, & 6)\, y_4 = t_2 \oplus x_4, & 7)\, t_3 = t_2 \oplus x_5, & 8)\, y_3 = t_3 \oplus x_3,\\
9)\, t_4 = t_3 \oplus x_4, & 10)\, y_2 = t_4 \oplus x_2, & 11)\, y_1 = t_4 \oplus x_3, & 12)\, y_8 = y_1 \oplus x_2.
\end{array}
$$

Now we obtain a new system of equations based on the equations given in (1).

$$
\begin{aligned}
y_1 &= x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8\\
y_2 &= x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8\\
y_3 &= x_3 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8\\
y_4 &= x_4 \oplus x_6 \oplus x_7 \oplus x_8\\
y_5 &= x_5 \oplus x_6 \oplus x_7 \oplus x_8\\
y_6 &= x_6 \oplus x_8\\
y_7 &= x_1 \oplus x_7\\
y_8 &= x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8
\end{aligned}
\qquad
\mathbf{y}^T = \mathbf{x} \cdot \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1\\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1\\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1\\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 1\\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1\\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1\\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1
\end{pmatrix} \tag{2}
$$

The implementation cost of (2), using the Paar algorithm, is 8 XOR.

$$
\begin{array}{llll}
1)\, y_7 = x_1 \oplus x_7, & 2)\, y_6 = x_6 \oplus x_8, & 3)\, t_1 = y_6 \oplus x_7, & 4)\, y_4 = t_1 \oplus x_4,\\
5)\, y_5 = t_1 \oplus x_5, & 6)\, y_3 = y_5 \oplus x_3, & 7)\, y_1 = y_3 \oplus x_4, & 8)\, y_2 = y_8 = y_1 \oplus x_2.
\end{array} \tag{3}
$$

Although (3) is not the implementation of (1), we can extend (3) to obtain an implementation with 10 XOR for the system of equations in (1). Therefore we have

$$
\begin{array}{lllll}
1)\, y_7 = x_1 \oplus x_7, & 2)\, y_6 = x_6 \oplus x_8, & 3)\, t_1 = y_6 \oplus x_7, & 4)\, y_4 = t_1 \oplus x_4, & 5)\, y_5 = t_1 \oplus x_5,\\
6)\, y_3 = y_5 \oplus x_3, & 7)\, y_1 = y_3 \oplus x_4, & 8)\, y_2 = y_8 = y_1 \oplus x_2, & 9)\, y_2 = y_2 \oplus x_3, & 10)\, y_5 = y_5 \oplus x_6.
\end{array}
$$

The main contribution of this paper is to find an answer to the following question: Let $\mathbf{A}$ be a binary matrix. Can some entries of $\mathbf{A}$ be modified so that the cost of implementing the new binary matrix is less than the cost of implementation of $\mathbf{A}$?

Although there are different SLP applications, we have used the Paar algorithm to perform our implementations. Using the Paar algorithm has two advantages. The first one is the speed of Paar algorithm, since it can be run on large binary matrices. The second advantage is related to the structure of this algorithm. In each round of Paar algorithm we get two columns such that have the maximum intersection between all possible choices. Now, by changing the zero entries of a binary matrix to 1, we expect the number of iterations of the Paar algorithm to decrease and reach to a low-cost implementation.

In addition, we define the concept of base matrix to construct the binary form of the MDS matrix. Although companion binary matrices are one of the important choices used to construct the binary form of MDS matrices, we show that low-cost implementations can be achieved through basic matrices.

1.1 Preliminaries

Let $\mathbf{A}$ be an $n \times n$ binary matrix. The number of required XOR to implement $\mathbf{A}$, using the Paar algorithm, is denoted with $\mathrm{CP}(\mathbf{A})$. For example, suppose the 8×8 binary matrix in (2) is called $\mathbf{A}$. Then it follows from (3) that $\mathrm{CP}(\mathbf{A}) = 8$.

Let $\mathbf{S}$ be a list. Then by $\mathrm{nops}(\mathbf{S})$ we mean the number of elements of $\mathbf{S}$. For instance, let $\mathbf{S} = [\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3]$ where $\mathbf{z}_1 = [[1, 2]]$, $\mathbf{z}_2 = [[7, 8]]$ and $\mathbf{z}_3 = [[5, 6], [9, 10]]$. Then we have $\mathrm{nops}(\mathbf{S}) = 3$, $\mathrm{nops}(\mathbf{z}_1) = \mathrm{nops}(\mathbf{z}_2) = 1$ and $\mathrm{nops}(\mathbf{z}_3) = 2$.

Let $\mathbb{F}_q$ denote the finite field containing q elements, where q is a prime power. An $n \times n$ matrix $\mathbf{A}$ is called MDS over $\mathbb{F}_q$ if any square submatrix of $\mathbf{A}$ is nonsingular [6]. MDS matrices are notable for their applications in the design of diffusion layers in block ciphers which can provide maximum diffusion. Recently, the Paar and BP algorithms have been used to implement MDS matrices in [5]. All our implementations are publicly available on GitHub:

`https://github.com/mousavi-codes/Implementation-of-Binary-Matrices`

2 Implementation of binary matrices

In this section, using the Paar algorithm, we propose a random-iterative method to implement binary matrices. Then, we introduce a heuristics algorithm for the implementation of MDS matrices. Also, using the heuristics algorithm, we implement a 8×8 involutory MDS matrix over 8-bit words with 408 XOR gates.

2.1 The Paar list method

First let's define the concept of a Paar list. In Definition 1, we consider a binary matrix $\mathbf{A}$ and then we select some zero entries of $\mathbf{A}$. Next, we change the selected entries of $\mathbf{A}$ to 1 and call it the new binary matrix $\mathbf{B}$. Finally, we compare $\mathrm{CP}(\mathbf{A})$ and $\mathrm{CP}(\mathbf{B})$.

Definition 1 Let $\mathbf{A} = (a_{i,j})$ be an $n \times m$ binary matrix and let r be the number of entries $\mathbf{A}$ equal to zero. Let $1 \leq p_1, p_2, \cdots, p_t \leq n$ and $1 \leq q_1, q_2, \cdots, q_t \leq m$ with $1 \leq t \leq r$ be positive integer such that $a_{p_k,q_k} = 0$ for all $1 \leq k \leq t$ and also $[p_i, q_i] \neq [p_j, q_j]$ for every $1 \leq i, j \leq t$ and $i \neq j$. Set $\mathbf{z} = [[p_1, q_1], [p_2, q_2], \cdots, [p_t, q_t]]$ and suppose that an $n \times m$ binary matrix $\mathbf{B_z} = (b_{i,j})$ with $1 \leq i \leq n$ and $1 \leq j \leq m$ is defined by

$$b_{i,j} = \begin{cases} 1 & [i,j] \text{ be an element of } \mathbf{z}, \\ a_{i,j} & \text{otherwise.} \end{cases} \tag{4}$$

If $\mathrm{CP}(\mathbf{B_z}) + \mathrm{nops}(\mathbf{z}) < \mathrm{CP}(\mathbf{A})$ then $\mathbf{z}$ is called a Paar list associated with the matrix $\mathbf{A}$.

Example 1 Let the finite field $\mathbb{F}_{2^4}$ be generated with the primitive polynomial $f(x) = x^4+x+1$ over $\mathbb{F}_2$. Let α be a root of $f(x)$. It can be checked that the following 8×8 Toeplitz matrix, given in [7], is an MDS matrix over $\mathbb{F}_{2^4}$.

$$\mathbf{T} = \begin{pmatrix} \alpha & 1 & \alpha^4 & 1 & \alpha^5 & \alpha^{14} & \alpha^7 & \alpha^8 \\ \alpha^3 & \alpha & 1 & \alpha^4 & 1 & \alpha^5 & \alpha^{14} & \alpha^7 \\ \alpha^6 & \alpha^3 & \alpha & 1 & \alpha^4 & 1 & \alpha^5 & \alpha^{14} \\ \alpha^{14} & \alpha^6 & \alpha^3 & \alpha & 1 & \alpha^4 & 1 & \alpha^5 \\ \alpha^{14} & \alpha^{14} & \alpha^6 & \alpha^3 & \alpha & 1 & \alpha^4 & 1 \\ \alpha^8 & \alpha^{14} & \alpha^{14} & \alpha^6 & \alpha^3 & \alpha & 1 & \alpha^4 \\ \alpha^6 & \alpha^8 & \alpha^{14} & \alpha^{14} & \alpha^6 & \alpha^3 & \alpha & 1 \\ \alpha^3 & \alpha^6 & \alpha^8 & \alpha^{14} & \alpha^{14} & \alpha^6 & \alpha^3 & \alpha \end{pmatrix}.$$

The binary form of the matrix $\mathbf{T}$, is a 32×32 binary matrix $\mathbf{A}$ that is given in Appendix A. In Table 1, some Paar lists related to the matrix $\mathbf{A}$ are obtained.

Definition 2 Let $\mathbf{A}$ be a binary matrix. Let $\mathbf{z}$ be a Paar list associated with $\mathbf{A}$. Suppose that the binary matrix $\mathbf{B_z}$ satisfies (4). Then, $\mathbf{z}$ is called an optimal Paar list if $\mathrm{CP}(\mathbf{B_z}) + \mathrm{nops}(\mathbf{z})$ is minimal over all possible choices of Paar lists that are associated with $\mathbf{A}$.

Based on Definition 1, there are $2^r - 1$ cases to construct Paar lists which implies that obtaining a Paar list by the exhaustive search is not possible when r is a large number. Therefore, we propose an algorithm in order to obtain a near-optimal Paar list.

There are two important points about Algorithm 1. In Step 8, we select the Paar lists that satisfy $\mathrm{nops}(\mathbf{z}_{j_i}) = m_2$. This condition is given because of some observations that are obtained from running of Algorithm 1. A potential tie in Algorithm 1 can occur in Step 9 which means we have no scale that based on the scales we can decide which elements of $\mathbf{T}$ should be chosen.

Example 2 In this example, we run Algorithm 1 over the binary matrix $\mathbf{A}$ in Appendix A. *First round:*

$$\mathbf{L} = [[4, 24], [5, 11], [14, 2], [16, 4], [18, 8], [18, 18], [19, 20], [32, 32]].$$

$\mathbf{z}$	$\mathrm{CP}(\mathbf{B_z})$	$\mathrm{CP}(\mathbf{B_z}) + \mathrm{nops}(\mathbf{z})$	$\mathrm{CP}(\mathbf{A})$
$\mathrm{nops}(\mathbf{z}) = 1$			
$[[4, 24]]$	203	204	205
$[[32, 32]]$	202	203	205
$\mathrm{nops}(\mathbf{z}) = 2$			
$[[18, 18], [18, 8]]$	201	203	205
$[[16, 4], [4, 24]]$	200	202	205
$\mathrm{nops}(\mathbf{z}) = 3$			
$[[16, 27], [32, 32], [19, 20]]$	199	202	205
$[[18, 27], [2, 11], [16, 4]]$	198	201	205
$\mathrm{nops}(\mathbf{z}) = 4$			
$[[18, 27], [16, 4], [5, 11], [4, 24]]$	197	201	205
$[[16, 27], [2, 11], [32, 32], [16, 4]]$	196	200	205
$\mathrm{nops}(\mathbf{z}) = 5$			
$[[16, 27], [2, 11], [16, 4], [5, 11], [4, 24]]$	195	200	205
$[[18, 27], [16, 27], [2, 11], [16, 4], [14, 2]]$	194	199	205

Table 1: Paar lists associated with the matrix **A**

It can be checked that $\mathrm{nops}(\mathbf{R}) = 247$ (please see the GitHub repository). Also $m_1 = 201$ and $m_2 = 3$. Hence, we have

$$\mathbf{T} = [[[16, 4], [14, 2], [4, 24]], [[32, 32], [16, 4], [14, 2]]].$$

We randomly select $\mathbf{z} = [[32, 32], [16, 4], [14, 2]]$ and then update the list $\mathbf{S}$ and the matrix $\mathbf{A}$.

Second round:

We get $\mathbf{L} = [[2, 11], [16, 27], [18, 27]]$ which results in

$$\mathbf{R} = [[[16, 27], [2, 11]], [[18, 27], [2, 11]], [[18, 27], [16, 27]], [[18, 27], [16, 27], [2, 11]]].$$

It can be verified that $m_1 = 196$ and $m_2 = 2$ which implies that $\mathbf{T} = [[16, 27], [2, 11]]$ and hence $\mathbf{z} = [[16, 27], [2, 11]]$. It follows from Step 10 that $\mathbf{S} = [[32, 32], [16, 4], [14, 2], [16, 27], [2, 11]]$. Now, we update the matrix $\mathbf{A}$ and run the third round. It can be checked that we can not find a Paar list $\mathbf{z}$ associated with the $\mathbf{A}$ such that $\mathrm{nops}(\mathbf{z}) = 1$. Therefore, Algorithm 1 terminates and then returns the list $\mathbf{S}$. The matrix $\mathbf{A_S}$ is given in Appendix B. We have $\mathrm{CP}(\mathbf{A_S}) + \mathrm{nops}(\mathbf{S}) = 194 + 5 = 199$. Therefore, Algorithm 1 reduced the implementation of the matrix $\mathbf{A}$, given in Appendix A, from 205 XOR to 199 XOR gates.

Note that if we cannot compute all possible Paar lists in Step 4 in Algorithm 1, we propose to obtain only short-length Paar lists (for example, say upto 5).

Example 3 Let the finite field $\mathbb{F}_{2^8}$ be generated with the primitive polynomial $f(x) = x^8 + x^4 + x^3 + x^2 + 1$ over $\mathbb{F}_2$. Let α be a root of $f(x)$. The next 8×8 Hadamard matrix, given in

Algorithm 1: Construction of a Near-Optimal Paar List by a Random-Iterative Method

```
Input  : A binary matrix $\mathbf{A}$ such that the number of zero entries of $\mathbf{A}$ is equal to $r$.
Output: A near-optimal Paar list associated with the matrix $\mathbf{A}$.
1  Set $\mathbf{S} = []$.
2  If there is at least a Paar list $\mathbf{z}$ associated with $\mathbf{A}$ such that nops($\mathbf{z}$) = 1 then go in
   Step 3 else return $\mathbf{S}$ end if.
3  Get $\mathbf{L} = [[p_1, q_1], [p_2, q_2], \cdots, [p_t, q_t]]$ with $t \leq r$ such that $[[p_i, q_i]]$ with $1 \leq i \leq t$ are Paar
     lists associated with the matrix $\mathbf{A}$.
4  Let $\mathbf{R} = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k]$, $k \leq 2^t - t - 1$, be all possible Paar lists that can be constructed
     from $\mathbf{L}$ provided that $\text{nops}(\mathbf{z}_i) \geq 2$ for $1 \leq i \leq k$.
5  If $\text{nops}(\mathbf{R}) \geq 1$ then go in Step 7 else go to Step 6 end if.
6  Set $\mathbf{z}_i = [[p_i, q_i]]$ for $1 \leq i \leq t$. Let $m = \mathbf{min}(\{\text{CP}(\mathbf{B}_{\mathbf{z}_i}) \mid i\})$ for $1 \leq i \leq t$. Choose one of
     the $\mathbf{z}_i$'s randomly, such as $\mathbf{z}$, provided that $\text{CP}(\mathbf{B}_{\mathbf{z}}) = m$ and go to Step 10.
7  Let $m_1 = \mathbf{min}(\{\text{CP}(\mathbf{B}_{\mathbf{z}_i}) + \text{nops}(\mathbf{z}_i) \mid i\})$ and $m_2 = \mathbf{min}(\{\text{nops}(\mathbf{z}_i) \mid i\})$ for $1 \leq i \leq k$.
8  Let $\mathbf{T} = [\mathbf{z}_{j_1}, \mathbf{z}_{j_2}, \cdots, \mathbf{z}_{j_q}]$ with $q \leq k$, be elements of $\mathbf{R}$ which satisfy the following two
     conditions: $\text{CP}(\mathbf{B}_{\mathbf{z}_{j_i}}) + \text{nops}(\mathbf{z}_{j_i}) = m_1$ and $\text{nops}(\mathbf{z}_{j_i}) = m_2$ for all $1 \leq i \leq q$.
9  Choose an element of $\mathbf{T}$ randomly and call it $\mathbf{z}$.
10 Add the entries of $\mathbf{z}$ to $\mathbf{S}$. Obtain $\mathbf{B}_z$ and update $\mathbf{A}$ by $\mathbf{A} = \mathbf{B}_z$ and then go to Step 2.
```

[8], is an involutory MDS matrix over $\mathbb{F}_{2^8}$ and to be called Khazad MDS matrix.

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha^{25} & \alpha^{2} & \alpha^{50} & \alpha^{26} & \alpha^{3} & \alpha^{238} & \alpha^{198} \\ \alpha^{25} & 1 & \alpha^{50} & \alpha^{2} & \alpha^{3} & \alpha^{26} & \alpha^{198} & \alpha^{238} \\ \alpha^{2} & \alpha^{50} & 1 & \alpha^{25} & \alpha^{238} & \alpha^{198} & \alpha^{26} & \alpha^{3} \\ \alpha^{50} & \alpha^{2} & \alpha^{25} & 1 & \alpha^{198} & \alpha^{238} & \alpha^{3} & \alpha^{26} \\ \alpha^{26} & \alpha^{3} & \alpha^{238} & \alpha^{198} & 1 & \alpha^{25} & \alpha^{2} & \alpha^{50} \\ \alpha^{3} & \alpha^{26} & \alpha^{198} & \alpha^{238} & \alpha^{25} & 1 & \alpha^{50} & \alpha^{2} \\ \alpha^{238} & \alpha^{198} & \alpha^{26} & \alpha^{3} & \alpha^{2} & \alpha^{50} & 1 & \alpha^{25} \\ \alpha^{198} & \alpha^{238} & \alpha^{3} & \alpha^{26} & \alpha^{50} & \alpha^{2} & \alpha^{25} & 1 \end{pmatrix}.$$

The binary form of the matrix $\mathbf{H}$ is a 64×64 binary matrix $\mathbf{A}$, given in [5]. It can be checked that $\text{CP}(\mathbf{A}) = 488$ (as reported in [5]). Now, we run Algorithm 1 over the matrix $\mathbf{A}$. Then, we obtain the following list at the first round.

$$\mathbf{L} = [[6, 22], [7, 36], [8, 37], [14, 30], [15, 44], [16, 45], [22, 6], [23, 37], [24, 45], [30, 14], [31, 45],$$

$$[32, 37], [37, 40], [38, 12], [45, 48], [46, 62], [53, 37], [55, 12], [61, 45], [62, 46], [63, 4], [63, 22]].$$

We have $\text{nops}(\mathbf{L}) = 22$ which implies that to obtain the list $\mathbf{R}$ in Step 4, we should construct $2^{22} - 23$ binary matrices and then obtain the implementation cost of these matrices by the Paar algorithm. Implementing computation takes a long time, since the binary matrices are 64×64. Therefore, using $\mathbf{L}$, we only get the Paar lists that are less than 5 in length. Given the limitations mentioned above, we obtained the next near-optimal Paar list by Algorithm 1.

$$\mathbf{S} = [[6, 22], [22, 6], [14, 30], [30, 14]].$$

It can be checked that $\text{CP}(\mathbf{A}_{\mathbf{S}}) + \text{nops}(\mathbf{S}) = 477 + 4 = 481$ which implies that Algorithm 1 reduced the implementation cost of Khazad MDS matrix from 488 XOR to 481 XOR gates (please see the GitHub repository to review the proposed implementation of the Khazad MDS matrix with 481 XOR gates).

2.2 The base matrix method

Let $\mathbf{M}$ be an MDS matrix over $\mathbb{F}_{2^n}$ such that the finite field $\mathbb{F}_{2^n}$ is constructed from the irreducible polynomial $f(x)$ over $\mathbb{F}_2$. It is proposed in [5] that to obtain the binary form of $\mathbf{M}$ we can use the binary companion matrix whose characteristic polynomial over $\mathbb{F}_2$ is $f(x)$. In this subsection, we intend to extend the method proposed in [5].

Definition 3 Let the finite field $\mathbb{F}_{2^n}$ be generated with the irreducible polynomial $f(x)$ of degree n over $\mathbb{F}_2$. Let $\mathbf{M}$ be an MDS matrix over $\mathbb{F}_{2^n}$. Let $\mathbf{N}$ be an $n \times n$ binary matrix such that its characteristic polynomial over $\mathbb{F}_2$ be $f(x)$. Suppose that the binary form of $\mathbf{M}$, using $\mathbf{N}$, is denoted with $\mathbf{A}^{\mathbf{N}}$. The binary matrix $\mathbf{N}$ is called an optimal-base matrix for the MDS matrix $\mathbf{M}$, if $\mathrm{CP}(\mathbf{A}^{\mathbf{N}})$ be minimal over all possible choices of $n \times n$ binary matrices whose characteristic polynomials over $\mathbb{F}_2$ are $f(x)$.

In this paper, we first obtain the binary form of an MDS matrix $\mathbf{M}$ over $\mathbb{F}_{2^n}$ using the companion matrix, denoted $\mathbf{C}$, and call it $\mathbf{A}^{\mathrm{C}}$. Then, we compute the implementation cost of $\mathbf{A}^{\mathrm{C}}$ through the Paar algorithm and we set $\mathrm{X} = \mathrm{CP}(\mathbf{A}^{\mathbf{C}})$. Next, we randomly search for an $n \times n$ binary matrix $\mathbf{N}$ such that $\mathbf{N}$ satisfies the following two conditions:

1) The characteristic polynomial of $\mathbf{N}$ over $\mathbb{F}_2$ is equal to $f(x)$. 2) $\mathrm{CP}(\mathbf{A}^{\mathbf{N}}) < \mathrm{X}$.

Now, we update the value of X using the relation $\mathrm{X} = \mathrm{CP}(\mathbf{A}^{\mathbf{N}})$ and iterate our search. Although the output of the proposed method is generally not an optimal result, we expect to achieve an almost optimal result after a few iterations.

It is clear that the two binary matrices $\mathbf{A}^{\mathrm{C}}$ and $\mathbf{A}^{\mathbf{N}}$ are not the same. But they can be considered as two similar binary representations of the MDS matrix $\mathbf{M}$. In fact, the characteristic polynomial of $\mathbf{N}$ is the irreducible polynomial $f(x)$ which implies that $f(x)$ is the minimal polynomial of $\mathbf{N}$ and hence the rational canonical form of $\mathbf{N}$ is equal to $\mathbf{C}$. In other words, the two $n \times n$ binary matrices $\mathbf{N}$ and $\mathbf{C}$ are similar.

Example 4 Let the finite field $\mathbb{F}_{2^8}$ be generated with the primitive polynomial $f(x) = x^8 + x^6 + x^3 + x^2 + 1$ over $\mathbb{F}_2$. Let α be a root of $f(x)$. It can be checked that the following 8×8 matrix is an involutory Hadamard MDS matrix over $\mathbb{F}_{2^8}$.

$$\mathbf{M} = \begin{pmatrix} 1 & \alpha^{23} & \alpha^{2} & \alpha^{46} & \alpha^{24} & \alpha^{3} & \alpha^{147} & \alpha^{83} \\ \alpha^{23} & 1 & \alpha^{46} & \alpha^{2} & \alpha^{3} & \alpha^{24} & \alpha^{83} & \alpha^{147} \\ \alpha^{2} & \alpha^{46} & 1 & \alpha^{23} & \alpha^{147} & \alpha^{83} & \alpha^{24} & \alpha^{3} \\ \alpha^{46} & \alpha^{2} & \alpha^{23} & 1 & \alpha^{83} & \alpha^{147} & \alpha^{3} & \alpha^{24} \\ \alpha^{24} & \alpha^{3} & \alpha^{147} & \alpha^{83} & 1 & \alpha^{23} & \alpha^{2} & \alpha^{46} \\ \alpha^{3} & \alpha^{24} & \alpha^{83} & \alpha^{147} & \alpha^{23} & 1 & \alpha^{46} & \alpha^{2} \\ \alpha^{147} & \alpha^{83} & \alpha^{24} & \alpha^{3} & \alpha^{2} & \alpha^{46} & 1 & \alpha^{23} \\ \alpha^{83} & \alpha^{147} & \alpha^{3} & \alpha^{24} & \alpha^{46} & \alpha^{2} & \alpha^{23} & 1 \end{pmatrix}.$$

Now consider the 8×8 binary matrix $\mathbf{C}$, given in (5). The matrix $\mathbf{C}$ is a companion matrix and its characteristic polynomial over $\mathbb{F}_2$ is $f(x)$. Also we have $\mathrm{CP}(\mathbf{A}^{\mathbf{C}}) = 505$.

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{5}$$

First, we set $\mathrm{X} = 505$ and then using the proposed method we obtain the binary matrix in (6) after a few iterations.

$$\mathbf{N} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{6}$$

The characteristic polynomial of $\mathbf{N}$ over $\mathbb{F}_2$ is $f(x)$. Further, we have $\mathrm{CP}(\mathbf{A}^{\mathbf{N}}) = 432$. Therefore, using the base matrix $\mathbf{N}$, we reduced the implementation cost of $\mathbf{M}$ from 505 XOR to 432 XOR gates.

2.3 The Matrix-List algorithm

In this subsection, we combine the proposed methods in Subsections 2.1 and 2.2 and we propose a heuristics algorithm and call it the Matrix-List algorithm. The goal of the Matrix-List algorithm is to achieve an implementation of an MDS matrix with low cost. The Matrix-List algorithm has two phases. The initial phase is to obtain an almost optimal base matrix and the final phase is to prepare a Paar list for the binary matrix obtained from the initial phase. The Matrix-List method is summarised in Algorithm 2.

```
Algorithm 2: The Matrix-List Algorithm
  Input  : An MDS matrix $\mathbf{M}$ over finite fields.
  Output: A low-cost implementation of $\mathbf{M}$.
  Get a near-optimal base matrix $\mathbf{N}$ for the MDS matrix $\mathbf{M}$.
  Obtain a near-optimal Paar list $\mathbf{S}$ associated with the matrix $\mathbf{A}^{\mathbf{N}}$.
  Return the implementation of the binary matrix $\mathbf{A}^{\mathbf{N}}_{\mathbf{S}}$ by the Paar algorithm.
```

Note that Algorithm 2 is a heuristic algorithm and we need to run it several times to get a desired implementation for an MDS matrix. In Example 5, we implement an 8×8 involutory Hadamard MDS matrix over 8-bit words with 408 XOR gates.

Example 5 Let the finite field $\mathbb{F}_{2^8}$ be generated with the primitive polynomial $f(x) = x^8 + x^7 + x^6 + x + 1$ over $\mathbb{F}_2$. Let α be a root of $f(x)$. The following 8×8 Hadamard matrix, given in [9], is an involutory MDS matrix over $\mathbb{F}_{2^8}$.

$$\mathbf{M} = \begin{pmatrix} 1 & \alpha & \alpha^{157} & \alpha^{253} & \alpha^{2} & \alpha^{155} & \alpha^{59} & \alpha^{254} \\ \alpha & 1 & \alpha^{253} & \alpha^{157} & \alpha^{155} & \alpha^{2} & \alpha^{254} & \alpha^{59} \\ \alpha^{157} & \alpha^{253} & 1 & \alpha & \alpha^{59} & \alpha^{254} & \alpha^{2} & \alpha^{155} \\ \alpha^{253} & \alpha^{157} & \alpha & 1 & \alpha^{254} & \alpha^{59} & \alpha^{155} & \alpha^{2} \\ \alpha^{2} & \alpha^{155} & \alpha^{59} & \alpha^{254} & 1 & \alpha & \alpha^{157} & \alpha^{253} \\ \alpha^{155} & \alpha^{2} & \alpha^{254} & \alpha^{59} & \alpha & 1 & \alpha^{253} & \alpha^{157} \\ \alpha^{59} & \alpha^{254} & \alpha^{2} & \alpha^{155} & \alpha^{157} & \alpha^{253} & 1 & \alpha \\ \alpha^{254} & \alpha^{59} & \alpha^{155} & \alpha^{2} & \alpha^{253} & \alpha^{157} & \alpha & 1 \end{pmatrix}.$$

First, the authors in [5] have obtained 430 XOR gates for the implementation of the matrix $\mathbf{M}$ by the Paar algorithm. Now we propose an implementation for $\mathbf{M}$ with 408 XOR gates using the Matrix-List algorithm. (our implementation is given in the GitHub repository).

As reported in [5], the implementation cost of $\mathbf{M}$, using the companion matrix and the Paar algorithm, is 430 XOR gates. Therefore, based on the Step 1 in Algorithm 2, we need to obtain a 8×8 binary matrix $\mathbf{N}$ such that the characteristic polynomial of $\mathbf{N}$ over $\mathbb{F}_2$ is $f(x)$ and also $\mathrm{CP}(\mathbf{A}^{\mathbf{N}})$ be less than 430. The best result we have is the following matrix.

$$\mathbf{N} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

It can be checked that $\mathrm{CP}(\mathbf{A}^{\mathbf{N}}) = 412$. Using Algorithm 1 on binary matrix $\mathbf{A}^{\mathbf{N}}$, the following Paar list is obtained.

$$\mathbf{S} = [[40, 8], [48, 16], [56, 24], [64, 32]].$$

We have $\mathrm{CP}(\mathbf{A}^{\mathbf{N}}_{\mathbf{S}}) = 404$ which implies that the MDS matrix $\mathbf{M}$ can be implemented with $\mathrm{CP}(\mathbf{A}^{\mathbf{N}}_{\mathbf{S}}) + \mathrm{nops}(\mathbf{S}) = 404 + 4 = 408$ XOR gates. The matrix $\mathbf{A}^{\mathbf{N}}_{\mathbf{S}}$ is given in Appendix C.

3 Conclusion

First, we introduced the concept of a Paar list that is related to a binary matrix. Then, we proposed the Paar list method to obtain a low-cost implementation for the binary matrices. Based on the proposed method, we introduced a random-iterative algorithm and we reduced the implementation cost of Khazad MDS matrix from 488 XOR to 481 XOR gates. Also, we showed that the implementation cost of an MDS matrix is related to the base matrix used to construct the binary form of the MDS matrix. Finally, we proposed the Matrix-List algorithm to implement MDS matrices and we obtained an implementation with low-cost XOR for a 8×8 involutory MDS matrix over 8-bit words. Although the Paar algorithm has been used in the given methods, other SLP applications can be used for the proposed algorithms.

References

[1] C. Paar, "Optimized arithmetic for Reed-Solomon encoders," *Proceedings of IEEE International Symposium on Information Theory*, pp. 250-250, Jun. 1997.

[2] J. Boyar, P. Matthews and R. Peralta, "Logic Minimization Techniques with Applications to Cryptology," *Journal of Cryptology*, vol. 26, no. 2, pp. 280-312, Apr. 2013.

[3] S. Banik, Y. Funabiki and T. Isobe, "More Results on Shortest Linear Programs," *International Workshop on Security*, pp. 109-128, Jul. 2019

[4] Q.Q. Tan and T. Peyrin, "Improved Heuristics for Short Linear Programs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, pp. 203-230, Nov. 2019.

[5] H. Kranz, G. Leander, K. Stoffelen and F. Wiemer, "Shorter Linear Straight-Line Programs for MDS Matrices," *IACR Transactions on Symmetric Cryptology*, vol. 2017, no. 4, pp. 188-211, Nov. 2017.

[6] F.J. MacWilliams and N.J.A. Sloane, "The Theory of Error Correcting Codes," *North-Holland*, 1977.

[7] S. Sarkar, and H. Syed, "Analysis of Toeplitz MDS matrices," *Australasian Conference on Information Security and Privacy*, vol. 10343, pp. 3-18, May. 2017.

[8] P. Barreto and V. Rijmen, "The Khazad Legacy-Level Block Cipher," *In Proceedings of the first open NESSIE Workshop*, Belgium, Nov. 2000.

[9] S.M. Sim, K. Khoo, F. Oggier and T. Peyrin, "Lightweight MDS Involution Matrices," *International Workshop on Fast Software Encryption*, vol. 9054, pp. 471-493, Mar. 2015.

Appendix A

$$\mathbf{A} = \left(\begin{array}{cccc|cccc|cccc|cccc|cccc|cccc|cccc|cccc}
0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0&1&1&1&1&0&0&1&1&0&1&1&0&1&0\\
1&0&0&1&0&1&0&0&1&1&0&1&0&1&0&0&1&0&1&0&0&0&1&0&1&0&1&1&0&1&1&1\\
0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1&0&1&0&1&1&0&1&1\\
0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1&0&1&0&0&1&0&1\\
\hline
0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0&1&1&1&1&0&0&1&1&0&1\\
0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1&0&1&0&0&1&0&1&0&0&0&1&0&1&0&1&1\\
0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1&0&1&0&1\\
1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1&0&1&0\\
\hline
0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0&1&1&1&1&0&0\\
0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1&0&1&0&0&1&0&1&0&0&0&1&0\\
1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1\\
1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0\\
\hline
1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0&1&1\\
0&0&1&0&0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1&0&1&0&0&1&0&1&0\\
0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1\\
1&0&0&0&1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0\\
\hline
1&1&0&0&1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0\\
0&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1&0&1&0&0\\
0&0&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0\\
1&0&0&0&1&0&0&0&1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1\\
\hline
1&0&1&0&1&1&0&0&1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1\\
0&1&1&1&0&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1\\
1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0\\
0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1\\
\hline
0&1&1&0&1&0&1&0&1&1&0&0&1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0\\
0&1&0&1&0&1&1&1&0&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0\\
1&0&1&0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0\\
1&1&0&1&0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1\\
\hline
0&1&0&0&0&1&1&0&1&0&1&0&1&1&0&0&1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1\\
0&1&1&0&0&1&0&1&0&1&1&1&0&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&1&0&0&1\\
0&0&1&1&1&0&1&0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0\\
1&0&0&1&1&1&0&1&0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&1&0&0&1&0&0&1&0
\end{array}\right)$$

Appendix B

$$\mathbf{A_S} = \left(\begin{array}{cccc|cccc|cccc|cccc|cccc|cccc|cccc|cccc}
0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0&1&1&1&1&0&0&1&1&0&1&1&0&1&0\\
1&0&0&1&0&1&0&0&1&1&1&1&0&1&0&0&1&0&1&0&0&0&1&0&1&0&1&1&0&1&1&1\\
0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1&0&1&0&1&1&0&1&1\\
0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1&0&1&0&0&1&0&1\\
\hline
0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0&1&1&1&1&0&0&1&1&0&1\\
0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1&0&1&0&0&1&0&1&0&0&0&1&0&1&0&1&1\\
0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1&0&1&0&1\\
1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1&0&1&0\\
\hline
0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0&1&1&1&1&0&0\\
0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1&0&1&0&0&1&0&1&0&0&0&1&0\\
1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0&1\\
1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1&0&0&0\\
\hline
1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0&1&1\\
0&1&1&0&0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1&0&1&0&0&1&0&1&0\\
0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&0&1\\
1&0&0&1&1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0&1&1&0&1&1&0\\
\hline
1&1&0&0&1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&1&0&0&0\\
0&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1&0&1&0&0\\
0&0&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1&0\\
1&0&0&0&1&0&0&0&1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1\\
\hline
1&0&1&0&1&1&0&0&1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1\\
0&1&1&1&0&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0&1&1&0&1\\
1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0&0&1&1&0\\
0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1&0&0&1&1\\
\hline
0&1&1&0&1&0&1&0&1&1&0&0&1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1&1&0&0&0\\
0&1&0&1&0&1&1&1&0&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&1&0&0&1&0&1&0&0\\
1&0&1&0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0&0&0&1&0\\
1&1&0&1&0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&1&0&0&1&0&0&1&0&0&0&0&1\\
\hline
0&1&0&0&0&1&1&0&1&0&1&0&1&1&0&0&1&1&0&0&0&1&1&0&0&1&0&0&0&0&0&1\\
0&1&1&0&0&1&0&1&0&1&1&1&0&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&1&0&0&1\\
0&0&1&1&1&0&1&0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&0&0&0&1&1&0&1&0&0\\
1&0&0&1&1&1&0&1&0&1&0&1&1&0&0&0&1&0&0&0&1&1&0&1&1&0&0&1&0&0&1&1
\end{array}\right)$$

Appendix C

$$\left(\begin{array}{cccccccc|cccccccc|cccccccc|cccccccc|cccccccc|cccccccc|cccccccc|cccccccc}
1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&1&0&0&0&1&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&0&1&0&0&1&1&0&0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&1&0&1&0&0&1&0&0&0&0&0&0&0&1&1&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1&0&0&0&0&0&1&0&1&1&0&0&0&0&1&0&0&0&1&0&0&0&1&0&1&1&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&1&1&0&0&0&0&1&0&0&0&1&1&0&0&0&0&0&1&0\\
\hline
0&0&0&0&0&1&1&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&1&1&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&1&0&1\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0\\
1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&1&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&1&0&1&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&1&1&1&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&1&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&1&0&0&1&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1\\
\hline
1&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&1&0&0&0\\
0&1&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&1\\
1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0\\
0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1\\
0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&1&1&0&1&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&0&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1&0&1&0&0&0&1&1&1&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&1&1&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&1&0&0&1&1&0\\
\hline
0&0&0&0&1&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0&0&1&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&1\\
1&0&0&0&0&0&1&0&0&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0\\
1&0&1&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&1\\
1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&1&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1\\
0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&1&0&1&0&0&1&1&0&0&0&0&1&0&0&0&0\\
\hline
0&0&0&0&0&1&0&1&0&0&1&0&1&0&0&0&1&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&0&1&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&0&0&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&1&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&1&1&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&1&0&1&0&0&1&0&0\\
0&1&0&0&0&1&0&1&1&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1&0&0&0&0&0&1&0&1&1&0&0&0&0&1&0&0\\
0&0&0&1&0&0&0&1&1&0&1&0&0&1&1&0&0&0&0&1&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&1&0&0\\
\hline
0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&1&1&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&1&1&0\\
1&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&1&0&0&0&0\\
0&0&0&1&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0\\
0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&0&0&0\\
0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&1&0&1&0&0&1&0&0&0&0&0&0&0&1&1&0\\
1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&1&1&1&0&0&0&0&0&1&1&1&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1\\
1&0&1&0&0&1&1&0&0&0&0&1&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&1\\
\hline
1&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&1&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\
0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&1&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&1&1&1&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1\\
0&0&0&1&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&1&0&1&0&0&1&1&0&0&1&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0\\
\hline
0&0&1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0&0&1&1&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&1&0&0&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&1&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&1&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&1&1&1&0&0&0&0&0&0&1&0\\
1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&1&0&1&0&0&1&1&0&0&0&0&1&0&0&0&1&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{array}\right)$$