# Designing Optimal Implementations of Linear Layers (Full Version)

Ruoxin Zhao[1,2], Baofeng Wu[1] and Rui Zhang[1]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering,
Chinese Academy of Sciences
zhaoruoxin@iie.ac.cn
[2] University of Chinese Academy of Sciences

**Abstract.** Linear layer is a fundamental primitive for computer science, electronic engineering, and telecommunication. The performance of a linear layer depends on two aspects: diffusion ability and implementation cost, where the latter is usually measured by the number of XORs needed to implement it. For many years, linear layers have been implemented by computing co-ordinates of the output independently. With this method, costs are determined only by the matrices representing linear layers. However, we note that the implementation cost of a given linear layer depends not only on its matrix but also on the ways with which we implement it. So, in this paper, we focus on another implementation method: modifying input vectors to output step by step (MIOSS). This method uses fewer XORs than previous methods do and needs no extra temporary register. Besides, this method makes the implementation cost of a linear layer same as that of its inverse. With the new implementation method, we first clarify the measurement of implementation cost and the optimal implementation procedure of linear layers. Here, "optimal" means using fewest XORs. Then, to find the optimal implementation procedure of a given linear layer, we construct a graph-theoretical model and transfer the problem to the shortest path problem in graph theory. Although there has been several algorithms for the shortest path problem, they do not perform best for the graph that we construct in this paper because of its regularity. Therefore, we adopt a new "double-direction" algorithm that uses less storage and makes the search for a shortest path more efficient in a regular graph. Unfortunately, this algorithm is not practical for large size linear layers because of its high space/time complexity. So, we finally construct another algorithm for finding efficient implementations of linear layers. An important advantage of this last algorithm is its extremely low complexity. We conduct it to the linear layer of AES and get very efficient implementations.

**Keywords:** Linear Layer · XOR Count · Equivalence Relation · Regular Graph · The Shortest Path

## 1 Introduction

Linear layer is a fundamental primitive for cryptography and is widely used in many aspects of computer science, electronic engineering and telecommunication. For example, it is used as diffusion layer in cryptography. Most modern block ciphers and hash functions are composed by confusion layers and diffusion layers. From the viewpoint of mathematics, confusion layers are usually nonlinear functions (S-boxes) while diffusion layers are usually linear functions. The goal of confusion layers is to increase chaos of information. And the goal of diffusion layers is to spread the chaos caused by confusion layers as much as possible. The performance of a linear layer depends on two aspects: diffusion ability and implementation efficiency. In general, these two aspects often contradict. In other words, the higher its diffusion ability is, the lower its implementation efficiency is. So, finding proper tradeoffs is a challenge for designers.

Essentially, linear layers are $\mathbb{F}_q$-linear transformations over $(\mathbb{F}_q)^n$. We only concern about $q$ as powers of 2 since they are almost all the cases in computer science. According to correspondence between linear transformations and matrices, every $\mathbb{F}_q$-linear layer can be represented by a matrix over $\mathbb{F}_q$. And this representation is unique under a fixed basis of $(\mathbb{F}_q)^n$. Some linear layers are just matrices over $\mathbb{F}_2$ such as the candidates presented in [9, 12, 7]. They are very convenient for implementation. However, there are also many linear layers over the extension fields of $\mathbb{F}_2$ such as the candidates presented in [4, 1, 8, 11]. A typical example of this type is the linear layer used in AES. It is a $(4 \times 4)$ matrix over $\mathbb{F}_{2^8}$. Note that multiplying a fixed element in $\mathbb{F}_{2^m}$ is actually an $\mathbb{F}_2$-linear transformation over $(\mathbb{F}_2)^m$ and can be represented by an $(m \times m)$ matrix over $\mathbb{F}_2$. So, an $\mathbb{F}_{2^m}$-linear layer over $(\mathbb{F}_{2^m})^n$ is actually an $\mathbb{F}_2$-linear transformation over $(\mathbb{F}_2)^{mn}$ and can be represented by an $(mn \times mn)$ matrix over $\mathbb{F}_2$. Hence, matrices over $\mathbb{F}_2$ are generic forms for linear layers.

Because every linear layer can be represented by a matrix over $\mathbb{F}_2$ and every element in $\mathbb{F}_{2^m}$ is represented by a bit string in computers, the number of bit-operations needed for implementing linear layers naturally becomes the measurement of implementation cost of them. In this work, we concentrate on how to implement a given linear layer with as few XORs as possible.

### 1.1 Related Work

As we mentioned above, the implementation efficiency of a linear layer is measured by the number of XORs needed to implement it. For an invertible $n \times n$ matrix $L$ over $\mathbb{F}_2$ and an input column vector $X$, $LX$ is usually implemented by computing the co-ordinates independently. For example, the first co-ordinate of $LX$ is computed as the product of the first row of $L$ and $X$. Consequently, the number of XORs of $L$ is naturally considered to be the difference between the Hamming weight of $L$ and the its order $n$. However, if a linear layer $L'$ is an invertible $n \times n$ matrix over $\mathbb{F}_{2^m}$, counting its number of XORs is a little complicated. In [11], the authors investigated this case in details. They formally defined the XOR counts of elements in $\mathbb{F}_{2^m}$ which is closely relevant to the computational pattern over $\mathbb{F}_{2^m}$, or more specifically, the basis of $\mathbb{F}_{2^m}$. Then, the XOR count of $L'$ is the sum of XOR counts of all entries of $L'$ plus $mn(n-1)$. In fact, according to our statement above, $L'$ can be alternatively represented by an $(mn \times mn)$ matrix $L''$ over $\mathbb{F}_2$ once we fix an $\mathbb{F}_2$-basis of $\mathbb{F}_{2^m}$. From this viewpoint, the XOR count of $L'$ defined in [11] is exactly the number of XORs needed to implement the $(mn \times mn)$ matrix $L''$, namely, the difference between the Hamming weight of $L''$ and its order $mn$. Later, some papers ([10, 8, 7]) about linear layers adopted the essentially same measurement of implementation cost as in [11]. Here we point out that all those papers have identical essence: computing the co-ordinates of outputs of a linear layer independently. Thus, the implementation cost of linear layers in those papers is only related to their matrices over $\mathbb{F}_2$ once bases are fixed.

Recently, Beierle et al. [1] presented a new measurement of implementation cost of multiplication with a fixed element in finite field $\mathbb{F}_{2^m}$. With the new measurement, they constructed a series of lightweight maximum distance separable (MDS) linear layers. In brief, their new measurement originally came from an implementation procedure different from previous ones. And this implementation procedure can indeed save XORs in comparison with previous methods.

## 1.2 Our Contributions

In this work, our goal is to find the optimal implementation for any given linear layer. Here, "optimal" means using fewest XORs. To attain the goal, we investigate the relation between implementation cost and implementation procedure. From the investigation, we present a generic measurement for implementation cost of linear layers. After that, we construct a graph-theoretical model and transfer the main problem to the shortest path problem in graph theory. Then, we construct a particular algorithm that is very proper to solve the particular shortest path problem related to linear layers. However, this algorithm for finding optimal implementations of linear layers is not practical because of its high space/time complexity. So, we construct another practical algorithm finally. This last algorithm has very low space/time complexity although we cannot guarantee that it necessarily gives optimal implementations of linear layers.

In Section 3, we firstly investigate the effect of different implementation procedures for implementation cost of linear layers. Briefly, we compare two implementation strategies: computing the co-ordinates of output vectors independently and modifying the input vectors to output step by step (MIOSS). The former is straightforward and has been used for years. But the latter need fewer XORs and no extra temporary register. Therefore, we focus on the latter one. From the investigation, we present a reasonable measurement of implementation cost of a linear layer $L$: the minimum number of additive elementary matrices in $L$'s factorization of the form $P_1A_1 \cdots P_sA_sP_{s+1}$ where every $P_i$ is a permutation matrix and every $A_j$ is an additive elementary matrix. Then, we give some properties about our measurement. These properties make our measurement more comprehensible.

In Section 4, we firstly give an equivalence relation over all invertible linear layers of order $n$ over $\mathbb{F}_2$. Based on this equivalence relation, we construct a graph whose vertices are all the equivalence classes. Then we show finding an optimal implementation of a given linear layer $L$ is essentially same as finding a shortest path between the vertex containing $L$ and the vertex containing the identity matrix $I_n$. Besides, we talk about some properties of the graph that are useful for solving the shortest path problem in the graph.

In Section 5, our main aim is to solve the shortest path problem in the graph defined in Section 4. Although there has already been "single-direction" methods (Dijkstra's algorithm, for example) for the shortest path problem, we abandon them and construct a "double-direction" algorithm. That is because the graph that we are talking about is a regular graph, and our double-direction algorithm uses less storage and makes the search for a shortest path more efficient in a regular graph. With our algorithm, we perform experiments to some linear layers and get good results.

Although the algorithm in Section 5 can give us optimal implementations of linear layers, it is not practical because of its high space/time complexity. To solve this problem, we finally present another algorithm in Section 6. An important advantage of this algorithm is that its space/time complexity is very low and we can run it on a common PC. On the other hand, we have to admit that it does not necessarily give us optimal implementations of linear layers. As an application, we use this algorithm to investigate the linear layer of AES and get implementations of it much more efficient than before.

## 2 Preliminary

### 2.1 Notations

In this paper, $\mathbb{F}_q$ or $GF(q)$ denotes the finite field of $q$ elements. $\mathcal{M}_{m\times n}(R)$ denotes the set consisting of all $(m \times n)$ matrices over a ring $R$, $\mathcal{IM}_{n\times n}(R)$ denotes the set consisting of all $(n \times n)$ invertible matrices over a ring $R$, $\mathcal{PM}_{n\times n}(R)$ denotes the set consisting of all $(n \times n)$ permutation matrices over a ring $R$, $\mathcal{EEM}_{n\times n}(R)$ denotes the set consisting of all $(n \times n)$ exchanging elementary matrices over a ring $R$, $\mathcal{MEM}_{n\times n}(R)$ denotes the set consisting of all $(n\times n)$ multiplicative elementary matrices over a ring $R$, and $\mathcal{AEM}_{n\times n}(R)$ denotes the set consisting of all $(n \times n)$ additive elementary matrices over a ring $R$. For a matrix $A$, $A^T$ denotes the transpose of $A$ and $W_H(A)$ denotes the Hamming weight (the number of nonzero entries) of $A$. $E_{i,j}$ denotes the matrix whose $(i, j)$th entry is 1 and other entries are 0. $I_n$ denotes the $(n \times n)$ identity matrix. For a set $S$, $\sharp S$ or $|S|$ denotes the cardinality of $S$.

## 2.2 Some Basic Facts about Linear Algebra

In this subsection, we review some basic facts about linear algebra. For more details, refer to [6].

To begin with, let us talk about elementary operations to matrices. For every matrix $L \in \mathcal{M}_{m\times n}(F)$ where $F$ is a field, there are three types of elementary row operations on it: row exchanging, row multiplication, and row addition. Row exchanging means exchanging tow rows of $L$. Row multiplication means multiplying every entries of a row by an invertible element of $F$. Row addition means adding a scalar-product of an element in $F$ and a row to another row. In addition to elementary row operations, there are also three types of corresponding elementary column operations. All the elementary operations to matrices are invertible.

Corresponding to elementary operations, there are three types of elementary matrices: exchanging elementary matrices, multiplicative elementary matrices, and additive elementary matrices. An $(n \times n)$ exchanging elementary matrix is the matrix obtained by exchanging two rows of $I_n$. An $(n \times n)$ multiplicative elementary matrix is the matrix obtained from $I_n$ by substituting an invertible element in $F$ for a 1 on the diagonal of $I_n$. An $(n \times n)$ additive elementary matrix $A_{i,j}$ is the matrix $I_n + aE_{i,j}$ where $a \in F$ and $i \neq j$. For any matrix $L \in \mathcal{M}_{m\times n}(F)$, left-multiplying $L$ by an $(m \times m)$ elementary matrix causes a corresponding elementary row operation to it, while right-multiplying $L$ by an $(n \times n)$ elementary matrix causes a corresponding elementary column operation to it. A useful fact is that every elementary matrix is invertible. The inverse matrix of an exchanging elementary matrix is just itself. And the inverse matrix of an additive elementary matrix $A_{i,j} = I_n + aE_{i,j}$ is $I_n - aE_{i,j}$.

Another type of matrices important for this paper is permutation matrix. An $n \times n$ permutation matrix is the matrix obtained from $I_n$ by permutes the rows of it. It is trivial that a matrix $P \in \mathcal{M}_{n\times n}(F)$ is a permutation matrix if and only if it is invertible and its entries are zero except $n$ entries equal to 1. For any matrix $L \in \mathcal{M}_{m\times n}(F)$, left-multiplying $L$ by an $(m \times m)$ permutation matrix permutes the rows of it, while right-multiplying $L$ by an $(n \times n)$ permutation matrix permutes the columns of it. In this paper, we sometimes write a permutation matrix $P \in \mathcal{M}_{n\times n}(F)$ as a column $(\rho(1), \cdots, \rho(n))^T$, where $\rho$ is a permutation over the set $N = \{1, \cdots, n\}$ and $\rho(i)$ is the column index of the nonzero entry in the $i$-th row of $P$ for $i = 1, \cdots, n$. For example, we write the permutation matrix

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} \rho(1) \\ \rho(2) \\ \rho(3) \\ \rho(4) \end{pmatrix}.$$

With this notation, if $P = (\rho(1), \cdots, \rho(n))^T$, then $P^{-1} = (\rho^{-1}(1), \cdots, \rho^{-1}(n))^T$ is also a permutation matrix, and the $i$-th row of $PL$ is the $\rho(i)$-th row of $L$.

## 2.3 Some Basic Facts about Graph Theory

In this subsection, we review some basic facts about graph theory. For more details, refer to [2, 5].

A graph $G$ is composed of two types of objects. It has a set $V$ of elements called vertices (or nodes) and a set of unordered pairs of vertices called edges. We denote the graph whose vertex set is $V$ and edge set is $E$ by $G = (V, E)$. The cardinality of the vertex set $V$ is called the order of the graph $G$. If $\alpha = \{x, y\}$ is an edge of $G$, we say that $\alpha$ joins $x$ and $y$, $x$ and $y$ are adjacent, and $x$ and $\alpha$ are incident. The degree of a vertex $x$ is a the number of edges that are incident with $x$ and is denoted by $\deg(x)$. If the degree of every vertex of the graph $G$ is $r$, we say $G$ is an $r$-regular graph.

In a graph $G = (V, E)$, a sequence of $m$ edges of the form

$$\{x_0, x_1\}, \{x_1, x_2\}, \cdots, \{x_{m-1}, x_m\}$$

is called a walk of length $m$. We also denote it by $x_0 - x_1 - \cdots - x_m$. The walk $x_0 - x_1 - \cdots - x_m$ is closed (open) if $x_0 = x_m$ ($x_0 \neq x_m$). If a walk has distinct edges, we call it a trail. In addition, if a trail has distinct vertices, we call it a path. A closed path is called a cycle. The distance between two vertices $x$ and $y$ is the shortest length of walks joining them and denoted by $d(x, y)$. It is clear that a walk joining $x$ and $y$ of length $d(x, y)$ is a path. We say two vertices $x$ and $y$ are connected if there exists a walk joining them. And we say a graph $G$ is connected if every pair of vertices of $G$ is connected.

### 2.4 Linear Layers

In general, an $F$-linear layer is an $F$-linear transformation over $F^n$ for a field $F$. In this paper, we merely focus on fields of characteristic 2 because they are widely used in computer science and telecommunication.

Suppose $L$ is an $\mathbb{F}_q$-linear layer over $(\mathbb{F}_q)^n$, where $q = 2^m$. According to the correspondence between linear transformations and matrices, $L$ can be uniquely represented by a matrix in $\mathcal{M}_{n\times n}(\mathbb{F}_q)$ under a given $\mathbb{F}_q$-basis of $(\mathbb{F}_q)^n$. Let

$$L = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,n} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_{n,1} & \alpha_{n,2} & \cdots & \alpha_{n,n} \end{pmatrix} \in \mathcal{M}_{n\times n}(\mathbb{F}_q).$$

Then for every input column $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)^T \in (\mathbb{F}_q)^n$, the output is $Y = LX = (\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_n)^T \in (\mathbb{F}_q)^n$, where $\boldsymbol{y}_i = \sum_{j=1}^n \alpha_{i,j}\boldsymbol{x}_j$ for $i = 1, \cdots, n$. Note that being multiplied by a fixed element in $\mathbb{F}_q$ is an $\mathbb{F}_2$-linear transformation over $\mathbb{F}_q$ and can be uniquely represented by a matrix in $\mathcal{M}_{m\times m}(\mathbb{F}_2)$ under a given $\mathbb{F}_2$-basis of $\mathbb{F}_q$. Therefore, $L$ is essentially an $\mathbb{F}_2$-linear transformation over $\mathbb{F}_2^{mn}$ and can be represented by a matrix

$$L = \begin{pmatrix} L_{1,1} & L_{1,2} & \cdots & L_{1,n} \\ L_{2,1} & L_{2,2} & \cdots & L_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ L_{n,1} & L_{n,2} & \cdots & L_{n,n} \end{pmatrix} \in \mathcal{M}_{mn\times mn}(\mathbb{F}_2),$$

where $L_{i,j} \in \mathcal{M}_{m\times m}(\mathbb{F}_2)$ for $i, j = 1, \cdots, n$. From this viewpoint, every $\mathbb{F}_q$-linear layer is essentially an $\mathbb{F}_2$-linear transformation and can be represented by a matrix over $\mathbb{F}_2$. That gives us a uniform pattern for linear layers.

## 3 Measurement of Implementation Costs of Linear Layers

In this section, we clarify how to measure the implementation costs of linear layers. As we mentioned above, considering invertible linear layers over $GF(2)$ suffices.

In general, the implementation cost of a given linear layer $L$ over $GF(2)$ is measured by the number of additions (XORs) needed to compute the output vector $LX$ for input

$X$. In [11], the authors formally presented a measurement of the number of XORs (XOR-count) of elements in $GF(2^m)$ as well as of whole diffusion layers. Later, some papers ([10, 8, 7]) adopted that measurement. We know that multiplication with a given element in $GF(2^m)$ can be represented by an $\mathbb{F}_2$-linear transformation over $\mathbb{F}_2^m$, namely, a matrix $L$ in $\mathcal{M}_{m\times m}(\mathbb{F}_2)$. From this viewpoint, their measurement XOR-count is exactly $W_H(L)-m$ since co-ordinates of output vector are computed independently. In fact, this notion has been used for years. However, the implementation cost of a given linear layer depends not only on the linear layer itself but also on ways by which we implement it. To show the effect of different ways on implementation costs, let us think about Example 1.

**Example 1.** Suppose

$$L = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \in \mathcal{IM}_{4\times 4}(\mathbb{F}_2)$$

is a linear transformation over $\mathbb{F}_2^4$. Then for every input vector $X = (x_1, x_2, x_3, x_4)^T \in GF(2)^4$, the corresponding output is

$$Y = LX = \begin{pmatrix} x_1 + x_2 + x_3 \\ x_2 \\ x_1 + x_2 + x_3 + x_4 \\ x_1 + x_2 \end{pmatrix}.$$

We may compute $Y$ with a common method which has been used for years. That is, we compute the co-ordinates of $Y$ independently. In this case, it costs $W_H(L)-4=6$ additions (XORs) over $GF(2)$. However, we can adopt another way to compute $Y$ – modifying the input $X$ to the output $LX$ step by step as the following procedure.

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \xrightarrow{1} \begin{pmatrix} x_1 + x_2 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \xrightarrow{1} \begin{pmatrix} x_1 + x_2 \\ x_2 \\ x_1 + x_2 + x_3 \\ x_4 \end{pmatrix}$$

$$\xrightarrow{1} \begin{pmatrix} x_1 + x_2 \\ x_2 \\ x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 + x_4 \end{pmatrix} \xrightarrow{0} \begin{pmatrix} x_1 + x_2 + x_3 \\ x_2 \\ x_1 + x_2 + x_3 + x_4 \\ x_1 + x_2 \end{pmatrix} = LX.$$

We compute $LX$ from $X$ through a series of operations, where the numbers above the arrows denotes the number of XORs needed for corresponding steps. For instance, the 2nd step costs only 1 XOR because we can add $x_1 + x_2$ (that is already computed on preceding steps and saved in the vector) to $x_3$. The last step uses 0 XOR because it merely permutes the co-ordinates. With this procedure, we can get $LX$ by using only 3 XORs. Moreover, we do not necessarily use extra temporary register for it.

Computing every co-ordinate independently is indeed easy to implement and has been used for years. However, it is not necessarily the optimal implementation according to Example 1. Here, "optimal" means "using fewest XORs". To find the optimal implementations of linear layers, we must take implementation procedures into account. We would like to give a lemma at first.

**Lemma 1.** *Suppose $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$. Then there exists a series of matrices $P_i \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ and $A_j \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$ such that $L = P_1A_1P_2A_2\cdots P_sA_sP_{s+1}$.*

*Proof.* As we know, $L$ can be transformed to its equivalent standard form through a series of elementary row/column operations. $L$'s equivalent standard form is $I_n \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$ because $L$ is invertible. According to the invertibility of elementary operations, $I_n$ can be transformed to $L$ through a series of elementary row/column operations. So, $L$ is equal to the product of a series of elementary matrices. Note that multiplicative elementary matrix over $GF(2)$ is just $I_n$ and can be omitted from the product. Besides, the product of some exchanging elementary matrices with order $n$ is a permutation matrix. Thus, $L$ is equal to the form $P_1A_1P_2A_2\cdots P_sA_sP_{s+1}$, where $P_i \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ for $i = 1, \cdots, s$, $A_j \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$ for $j = 1, \cdots, s + 1$. □

In accordance with Lemma 1, for every linear layer $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$ and input vector $X = (x_1, \cdots, x_n)^T \in \mathbb{F}_2^n$, the output vector is $LX = P_1A_1P_2A_2\cdots P_sA_sP_{s+1}X$ which means we can get the output vector through a series of co-ordinate permutations or additive elementary operations. On the contrary, every modification procedure from $X$ to $LX$ can be expressed by the form $P_1A_1P_2A_2\cdots P_sA_sP_{s+1}X$ as in Example 1. Note that for implementation of $P_1A_1P_2A_2\cdots P_sA_sP_{s+1}X$, every $P_j$ costs 0 XOR while every $A_i$ costs 1 XOR. Therefore, $L$'s factorization $P_1A_1P_2A_2\cdots P_sA_sP_{s+1}$ in Lemma 1 shows the number of XORs used for its implementation. We adopt this form as the measurement of implementation costs of linear layers.

**Definition 1.** For a linear layer $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$, its minimum XOR-count $Min-XOR-Count(L)$ is the minimum number of additive elementary matrices $A_i$ such that $L$ can be written as form $P_1A_1P_2A_2\cdots P_sA_sP_{s+1}$, where every $P_i \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ and every $A_j \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$. And we call an implementation procedure $LX = P_1A_1P_2A_2\cdots P_rA_rP_{r+1}X$ containing $Min - XOR - Count(L)$ additive elementary matrices a minimum-XOR-implementation of $L$.

In addition to Definition 1, we have other ways to describe the optimal implementation procedures of linear layers. To clarify those ways, we need the following lemma.

**Lemma 2.** *Suppose* $P = (\rho(1), \cdots, \rho(n))^T \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$*,* $A_{r,s} \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$*. Then* $PA_{r,s} = A_{\rho^{-1}(r),\rho^{-1}(s)}P$ *where* $A_{\rho^{-1}(r),\rho^{-1}(s)} \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$*.*

*Proof.* As we mentioned before, left-multiplying $P$ permutes the rows of $A_{r,s}$. In detail, the $i$-th row of $PA_{r,s}$ is the $\rho(i)$-th row of $A_{r,s}$. So, the $\rho^{-1}(r)$-th row of $PA_{r,s}$ is the $r$-th row of $A_{r,s}$, and the $\rho^{-1}(s)$-th row of $PA_{r,s}$ is the $s$-th row of $A_{r,s}$. Obviously, $PA_{r,s}$ will become $P$ if we add $\rho^{-1}(s)$-th row of $PA_{r,s}$ to $\rho^{-1}(r)$-th row of it. Thus, $A_{\rho^{-1}(r),\rho^{-1}(s)}PA_{r,s} = P$. Then we get $PA_{r,s} = (A_{\rho^{-1}(r),\rho^{-1}(s)})^{-1}P = A_{\rho^{-1}(r),\rho^{-1}(s)}P$ since $(A_{\rho^{-1}(r),\rho^{-1}(s)})^{-1} = A_{\rho^{-1}(r),\rho^{-1}(s)}$. □

Combining Lemma 1 and Lemma 2, we get the next lemma easily. We omit its proof because it is really trivial.

**Lemma 3.** *Every* $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$ *can be factorized to the form* $L = \prod_{i=1}^{s} B_i$ *where every* $B_i$ *is either in* $\mathcal{PM}_{n\times n}(\mathbb{F}_2)$ *or in* $\mathcal{AEM}_{n\times n}(\mathbb{F}_2)$*.*

We call every factorization of $L$ as the form in Lemma 3 a P-AE factorization of it. Obviously, the factorization form in Lemma 1 is a P-AE factorization. On the contrary, every P-AE factorization can be written as the form in Lemma 1 since the product of permutation matrices in $\mathcal{PM}_{n\times n}(\mathbb{F}_2)$ is a permutation matrix too. Therefore, the minimum XOR count of a given linear layer $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$ is exactly equal to the minimum number of additive elementary matrices in $L$'s P-AE factorizations. In summary of this paragraph, we present the following theorem.

**Theorem 1.** *For every linear layer* $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$*,* $Min - XOR - Count(L)$ *is equal to the minimum number of additive elementary matrices in* $L$*'s P-AE factorizations.*

In [1], the authors measured the lowest implementation cost (it is called XOR count in [1]) of a linear layer $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$ by the minimum number $t$ such that $L$ can be written as $L = P\prod_{i=1}^{t} A_i$ where $P \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ and every $A_i \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$. It is easy to see that every factorization of the form $L = P\prod_{i=1}^{t} A_i$ corresponds to a factorization of the form in Definition 1 according to Lemma 2. So, our definition of Min-XOR-count does not contradict theirs. We shall indicate the advantage of our definition later.

Because a linear layer is often used bidirectionally (for example encryption and decryption), we also need to consider the inverse of it.

**Theorem 2.** *For every linear layer $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$, $Min-XOR-Count(L^{-1})$ is equal to $Min-XOR-Count(L)$. Moreover, if a minimum-XOR-implementation of $L$ is $P_1A_1P_2A_2\cdots P_rA_rP_{r+1}$, then $P_{r+1}^{-1}A_rP_r^{-1}\cdots A_1P_1^{-1}$ is a minimum-XOR-implementation of $L^{-1}$.*

*Proof.* For every factorization $L = Q_1B_1\cdots Q_sB_sQ_{s+1}$ where each $Q_i \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ and each $B_j \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$, we have a factorization $L^{-1} = Q_{s+1}^{-1}B_s^{-1}Q_s^{-1}\cdots B_1^{-1}Q_1^{-1}$. Note that the inverse of a permutation matrix is a permutation too, and the inverse of each additive elementary matrix $B_j$ is just $B_j$ itself. So, $Min-XOR-Count(L^{-1})$ is equal to $Min-XOR-Count(L)$. And the second part of the theorem is trivial. □

At the end of this section, we point out that if a linear layer $L'$ is a power of another linear layer $L$, for example $L' = L^5$, then minimum-XOR-implementation of $L'$ is better successive 5 times minimum-XOR-implementations of $L$. That is because successive 5 times minimum-XOR-implementations of $L$ is merely an implementation procedure of $L'$ and it cannot perform better than minimum-XOR-implementation of $L'$. Consequently, if $L' = L^r$, then $Min-XOR-Count(L') \leq rMin-XOR-Count(L)$.

## 4 A Graph-Theoretical Model

After clarifying the measurement of the lowest implementation cost of linear layers, we are confronted by two problems:

1. How to calculate the Min-XOR-Count of a given linear layer $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$;
2. How to get a minimum-XOR-implementation of a given linear layer $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$.

Obviously, the 1st problem will be trivial provided that we solve the 2nd one. In this section, we would like to handle the two problems by a graph-theoretical model.

First of all, we introduce a relation over $\mathcal{IM}_{n\times n}(\mathbb{F}_2)$. We say two matrices $B, C \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$ are row-permutation-equivalent (denoted by $B \sim_{RP} C$) if there exists a $Q \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ such that $B = QC$. For every $B \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$, $B \sim_{RP} B$ since $B = I_nB$. If $B \sim_{RP} C$, then $C \sim_{RP} B$ because $C = Q^{-1}B$ and $Q^{-1} \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ too. If there exists $Q_1, Q_2 \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ such that $B = Q_1C$ and $C = Q_2D$, then $B = Q_1Q_2D$ and $Q_1Q_2 \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$. Therefore, "$\sim_{RP}$" is an equivalence relation over $\mathcal{IM}_{n\times n}(\mathbb{F}_2)$. For every $B \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$, let $[B]_{RP}$ denote the equivalence class containing $B$ under "$\sim_{RP}$". It is easy to see that for every $B \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$, $[B]_{RP} = \{QB|Q \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)\}$, and $Q_1B \neq Q_2B$ if $Q_1 \neq Q_2$.

Next, we define a graph dependent on row-permutation-equivalence over $\mathcal{IM}_{n\times n}(\mathbb{F}_2)$.

**Definition 2.** For a positive integer $n$, let $G(n) = (V, E)$ be a graph where the vertex set consists of all the equivalence classes under row-permutation-equivalence relation over $\mathcal{IM}_{n\times n}(\mathbb{F}_2)$. And two vertices $[B]_{RP}$ and $[C]_{RP}$ are adjacent if there exists $B' \in [B]_{RP}$, $C' \in [C]_{RP}$ and $A \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$ such that $B' = AC'$.

Now let us show some useful properties of the graph in Definition 2.

**Theorem 3.** *Let $G(n) = (V, E)$ be the graph described in Definition 2 and $[B]_{RP}$ be a vertex of $G(n)$. Then $G(n)$ is an $(n^2 - n)$-regular graph, $[A_1B]_{RP} \neq [A_2B]_{RP}$ for distinct $A_1, A_2 \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$, and $[AB]_{RP}$ runs all the vertices adjacent to $[B]_{RP}$ when $A$ runs over $\mathcal{AEM}_{n\times n}(\mathbb{F}_2)$.*

*Proof.* If $[A_1B]_{RP} = [A_2B]_{RP}$, then $A_1B = PA_2B$ for some $P \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$. Consequently, we get $A_1 = PA_2$ by right-multiplying two sides of $A_1B = PA_2B$ by $B^{-1}$. We assert that $P$ must be $I_n$. Otherwise, some entries on the diagonal of $PA_2$ would be 0 and $PA_2$ cannot be equal to $A_1$. Then $A_1 = A_2$. So, $[A_1B]_{RP} \neq [A_2B]_{RP}$ for distinct $A_1, A_2 \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$.

On the other hand, if a vertex $[C]_{RP}$ is adjacent to $[B]_{RP}$, there exists $B' \in [B]_{RP}$, $C' \in [C]_{RP}$ and $A \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$ such that $B' = AC'$. Then $AB' = C'$ since $A^{-1} = A$. Consequently, there exists $P_2 \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ and $A' \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$ such that $C' = AB' = AP_1B = P_2A'B$ according to Lemma 2. So, $[C]_{RP} = [C']_{RP} = [P_2A'B]_{RP} = [A'B]_{RP}$. That shows $[AB]_{RP}$ runs all the vertices adjacent to $[B]_{RP}$ when $A$ runs over $\mathcal{AEM}_{n\times n}(\mathbb{F}_2)$.

In summary of two preceding paragraphs, the degree of every vertex of $G(n)$ is the cardinality of $\mathcal{AEM}_{n\times n}(\mathbb{F}_2)$, namely, $n^2 - n$. □

Finally, we present a significant theorem that explains why we set up the graph-theoretical model.

**Theorem 4.** *Let $G(n) = (V, E)$ be the graph described in Definition 2 and $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$. Then the minimum XOR-count of $L$ is equal to the distance between vertices $[L]_{RP}$ and $[I_n]_{RP}$ in $G(n)$.*

*Proof.* For every factorization of $L$ with the form $P_1A_1P_2A_2 \cdots P_sA_sP_{s+1}$, where every $P_i \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ and every $A_j \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$, there exists a path

$$\begin{aligned}&[L]_{RP} = [P_1A_1P_2A_2 \cdots P_sA_sP_{s+1}]_{RP} - [P_2A_2 \cdots P_sA_sP_{s+1}]_{RP}\\ &- \cdots - [P_sA_sP_{s+1}]_{RP} - [P_{s+1}]_{RP} = [I_n]_{RP}\end{aligned}$$

between $[L]_{RP}$ and $[I_n]_{RP}$ of length $s$.

On the other hand, for every path

$$[L]_{RP} - [L_{r-1}]_{RP} - \cdots - [L_2]_{RP} - [L_1]_{RP} - [I_n]_{RP}$$

of length $r$, according to Theorem 3, there must be $L = Q_{r-1}A_{r-1}L_{r-1}$, $L_j = Q_{j-1}A_{j-1}L_{j-1}$ for $j = 2, \cdots, r-1$, and $L_1 = Q_0A_0I_n$ for some $Q_i \in \mathcal{PM}_{n\times n}(\mathbb{F}_2)$ and $A_i \in \mathcal{AEM}_{n\times n}(\mathbb{F}_2)$ for $i = 0, 1, \cdots, r$. Consequently, $L = Q_{r-1}A_{r-1} \cdots Q_1A_1Q_0A_0$ which is a factorization of $L$ with the form in Lemma 1.

Therefore, the minimum XOR-count of $L$ is equal to the minimum length of paths between $[L]_{RP}$ and $[I_n]_{RP}$, namely, their distance. □

From the proof of Theorem 4, we can easily see that a shortest path between $[L]_{RP}$ and $[I_n]_{RP}$ indicates a minimum-XOR-implementation of $L$.

## 5 Searching for Optimal Implementations of Linear Layers

In this section, we talk about how to get a minimum-XOR-implementation of a given linear layer $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$. In accordance with the preceding section, this question is equivalent to finding a shortest path between $[L]_{RP}$ and $[I_n]_{RP}$.

Let us sketch our strategy firstly. Suppose $G(V, E)$ is a connected graph, $a, b \in V$. To find a shortest path between $a$ and $b$, we let $\mathcal{A}_0 = \{a\}$, $\mathcal{B}_0 = \{b\}$ and check whether $a = b$. If $a = b$, we do not have to do anything. If $a \neq b$, we construct a set $\mathcal{A}_1$ consisting of the vertices of $G$ that are adjacent to $a$. In other words, $\mathcal{A}_1$ consists of the vertices having distance 1 from $a$. Then we check whether there exists $x \in \mathcal{A}_1$ such that $x = b$. If there is, we get a shortest path $a - b$ between $a$ and $b$. If there is not, we construct a set $\mathcal{B}_1$ consisting of the vertices of $G$ that are adjacent to $b$. In other words, $\mathcal{B}_1$ consists of the vertices having distance 1 from $b$. Then we check whether there exists $x \in \mathcal{A}_1$ and $y \in \mathcal{B}_1$ such that $x = y$. If there is, we get a shortest path $a - a_1^* - b$ between $a$ and $b$ where $a_1^* \in \mathcal{A}_1$. If there is not, we proceed the procedure above: check and move forwards one step from $\mathcal{A}_i$, then check and move forwards one step from $\mathcal{B}_i$. Finally, we will find $x \in \mathcal{A}_{k+1}$ and $y \in \mathcal{B}_{k+1}$ (or $y \in \mathcal{B}_k$) for some $k$ such that $x = y$. As a result, we find a shortest path

$$a_0^* - a_1^* - \cdots - a_k^* - a_{k+1}^* - b_k^* - \cdots - b_1^* - b \ (or\ a - a_1^* - \cdots - a_k^* - b_k^* - \cdots - b_1^* - b_0^*)$$

between $a$ and $b$, where $a_i^* \in \mathcal{A}_i$, $b_i^* \in \mathcal{B}_i$, every pair $a_i^*, a_{i+1}^*$ and every pair $b_j^*, b_{j+1}^*$ are adjacent. We formally describe the above procedures in Algorithm 1 with pseudocode.

**Algorithm 1** Double-Direction Search for a Shortest Path

```
Require: a graph $G = (V, E)$, $a, b \in V$.
Ensure: a shortest path between $a$ and $b$.
  $\mathcal{A}_0 \leftarrow \{a\}$, $\mathcal{B}_0 \leftarrow \{b\}$, $i \leftarrow 0$, $j \leftarrow 0$, $link \leftarrow 0$;
  while $link = 0$ do
    if there exists $a_i^* \in \mathcal{A}_i$ and $b_j^* \in \mathcal{B}_j$ such that $a_i^* = b_j^*$ then
      $link \leftarrow 1$;
      continue;
    else
      $i \leftarrow i + 1$;
      construct a set $\mathcal{A}_i$ consisting of the vertices adjacent to some vertex in $\mathcal{A}_{i-1}$ and not in $\bigcup_{k=0}^{i-1} \mathcal{A}_k$;
    end if
    if there exists $a_i^* \in \mathcal{A}_i$ and $b_j^* \in \mathcal{B}_j$ such that $a_i^* = b_j^*$ then
      $link \leftarrow 1$;
      continue;
    else
      $j \leftarrow j + 1$;
      construct a set $\mathcal{B}_j$ consisting of the vertices adjacent to some vertex in $\mathcal{B}_{j-1}$ and not in $\bigcup_{k=0}^{j-1} \mathcal{B}_k$;
    end if
  end while
  return the path $a_0^* - \cdots - a_i^* - b_{j-1}^* - \cdots - b_0^*$ such that every pair $a_k^*, a_{k+1}$ and every pair $b_l^*, b_{l+1}^*$ are adjacent;
```

**Lemma 4.** *Suppose $G = (V, E)$ is a graph, $a, b \in V$. Then we get a shortest path between $a$ and $b$ with Algorithm 1.*

*Proof.* Assume there is a path $a - c_1 - \cdots - c_{r-1} - b$ shorter than the output of Algorithm 1. Then $c_k \in \mathcal{A}_k$ for $k = 1, \cdots, \lceil \frac{r-1}{2} \rceil$, and $c_{r-l} \in \mathcal{B}_l$ for $l = 1, \cdots, \lfloor \frac{r-1}{2} \rfloor$. Consequently, there exists $a_k^* \in \mathcal{A}_k$, $b_l^* \in \mathcal{B}_l$ such that $a_k^* = b_l^*$ for some $k < i$ or some $l < j$ which contradicts Algorithms 1. □

Algorithm 1 is a generic method for any graph. Now we use it to handle our main target: a minimum-XOR-implementation of a given linear layer $L \in \mathcal{IM}_{n \times n}(\mathbb{F}_2)$. For this target, we present Algorithm 2 and omit the proof of correctness because it directly comes from Algorithm 1. Here, we just give an explanation of Algorithm 2 as follows.

- We choose an element in every equivalence class to represent it. For example, $L_i^*$ represents the class $[L_i^*]_{RP}$. Hence, we determine $[L_i]_{RP} = [B_j]_{RP}$ by checking $L_i \sim_{RP} B_j$.
- When we need to determine whether two invertible matrices $L_i$ and $B_j$ are row-permutation-equivalent, we check the Hamming weight of $L_i B_j^{-1}$ since $L_i \sim_{RP} B_j$

if and only if $W_H(L_i B_j^{-1}) = n$. This method can be implemented easily when $B$ is a product of some additive elementary matrices. More explicitly, if $B = A_1 \cdots A_s$, $B^{-1} = A_s \cdots A_1$.

**Algorithm 2** Finding a Minimum-XOR-Implementation of a Linear Layer

```
Require: a positive integer $n$, a matrix $L \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$.
Ensure: a minimum-XOR-implementation of $L$.
  $\mathcal{L}_0 \leftarrow \{L\}$, $\mathcal{B}_0 \leftarrow \{I_n\}$, $i \leftarrow 0$, $j \leftarrow 0$, $link \leftarrow 0$;
  while $link = 0$ do
    if there exists $L_i^* \in \mathcal{L}_i$ and $B_j^* \in \mathcal{B}_j$ such that $L_i^* \sim_{RP} B_j^*$ then
      $link \leftarrow 1$;
      continue;
    else
      $i \leftarrow i + 1$;
      construct a set $\mathcal{L}_i$ consisting of the matrices having the form $A_{r,s}L_{i-1}$ and not row-permutation-
      equivalent to any matrix in $\bigcup_{k=0}^{i-1}\mathcal{L}_k$, where $A_{r,s}$ runs over $\mathcal{AEM}_{n\times n}(\mathbb{F}_2)$ and $L_{i-1}$ runs over
      $\mathcal{L}_{i-1}$;
    end if
    if there exists $L_i^* \in \mathcal{L}_i$ and $B_j^* \in \mathcal{B}_j$ such that $L_i^* \sim_{RP} B_j^*$ then
      $link \leftarrow 1$;
      continue;
    else
      $j \leftarrow j + 1$;
      construct a set $\mathcal{B}_j$ consisting of the matrices having the form $A_{r,s}B_{j-1}$ and not row-permutation-
      equivalent to any matrix in $\bigcup_{k=0}^{j-1}\mathcal{B}_k$, where $A_{r,s}$ runs over $\mathcal{AEM}_{n\times n}(\mathbb{F}_2)$ and $B_{j-1}$ runs over
      $\mathcal{B}_{i-1}$;
    end if
  end while
  return the path $L_0^* - \cdots - L_i^* - B_{j-1}^* - \cdots - B_0^*$ such that every pair $L_k^*, L_{k+1}$ and every pair $B_l^*, B_{l+1}^*$
  are adjacent;
```

For a given graph and two vertices of it, there has already been algorithms in literature for finding the shortest path of them. For example, the famous Dijkstra's algorithm ([3], Chapter 11, page 443). Certainly, we can adopt it to solve the main problem in this paper. Essentially, Dijkstra's algorithm is a single-direction search, while our algorithms are double-direction search. Let us show what will happen if we handle our main problem with Dijkstra's algorithm. We use the same notations as that in Algorithm 1. We start from vertex $a$ to find a shortest path between $a$ and $b$. By means of Dijkstra's algorithm, we need to construct a series of sets $\mathcal{A}_i$ consisting of vertices whose distance to $a$ is $i$ for $i = 1, 2, \cdots$. According to Theorem 3, $|\mathcal{A}_1| = n^2 - n$, and $|\mathcal{A}_i| = (n^2 - n)(n^2 - n - 1)^{i-1}$ for $i \geq 2$ in the worst case. If the distance between $a$ and $b$ is $s \geq 2$, the algorithm will not terminate until $\mathcal{A}_s$ is constructed. We see the cardinality of $\mathcal{A}_i$s increase too fast and will occupy too much space. Actually, that is the reason why we abandon single-direction strategies and adopt a double-direction strategy – moving forwards step by step from $a$ and $b$ alternately. With a double-direction strategy, we can save a lot of memory space and modify the search for the shortest path. For instance, suppose the distance between $a$ and $b$ is $s = 2t$. If we use our method, then we need to construct $\mathcal{A}_i, \mathcal{B}_i$ for $i = 1, \cdots, t$, where $|\mathcal{A}_1| = |\mathcal{B}_1| = n^2 - n$, $|\mathcal{A}_2| = |\mathcal{B}_2| = (n^2-n)(n^2-n-1)$,$\cdots$,$|\mathcal{A}_t| = |\mathcal{B}_t| = (n^2-n)(n^2-n-1)^{t-1}$ in the worst case. However, if we use a single-direction strategy, we have to construct $\mathcal{A}_i$ for $i = 1, \cdots, 2t$, where $|\mathcal{A}_1| = n^2-n$, $|\mathcal{A}_2| = (n^2-n)(n^2-n-1)$,$\cdots$,$|\mathcal{A}_t| = (n^2-n)(n^2-n-1)^{t-1}$,$|\mathcal{A}_{t+1}| = (n^2-n)(n^2-n-1)^t$,$\cdots$,$|\mathcal{A}_{2t}| = (n^2-n)(n^2-n-1)^{2t-1}$.

## 5.1 Experimental Results

We search the minimum XOR implementations of many linear layers and list some of them in Appendix A. The optimal implementation procedure of each one is like what we show in Example 1. Meanwhile, we list the minimum XOR count and the difference between Hamming weight and the order of the linear layers in Table 1, where the former indicates implementation cost of our strategy and the latter indicates the cost of previous implementation methods.

## 6 A More Practical Strategy for Efficient Implementations of Linear Layers

Although we present an algorithm to search for an optimal implementation of a given linear layer, its space/time complexity skyrockets along with the increase of order and minimum XOR count of the given linear layer. For example, in the case when a linear layer $L$ is in $\mathcal{IM}_{8\times 8}(\mathbb{F}_2)$, the time complexity of Algorithm 2 is $56^r$, where $r = Min{-}XOR{-}Count(L)$. If $r = 15$ (not very large), then the time complexity of searching for the minimum XOR implementation of $L$ will be $56^{15} \approx 2^{87}$.

In this section, to avoid high computational complexity of searching for optimal implementations of linear layers, we switch to other efficient implementations of them. Our aim is still looking for a P-AE factorization of a given linear layer $M$, but it is not necessarily a minimum XOR implementation of it. The guideline is trying to reduce the Hamming weight of the given linear layer as much as possible by an additive row/column elementary operation on each step. Let us give Algorithm 3 first.

**Algorithm 3** Finding an Efficient Implementation of a Linear Layer

```
Require: a positive integer $n$, a matrix $M \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$;
Ensure: a series of additive elementary operations and a permutation matrix;
  $r \leftarrow 0$, $RUB \leftarrow W_H(M) - n$;
  while $W_H(M) > n$ and $r \leq RUB$ do
    let $\alpha_i$ denote the $i$-th row of $M$ and $\beta_i$ denote the $i$-th column of $M$ for $i = 1, \cdots, n$, $weightdecrease \leftarrow$
    $W_H(\alpha_1) - W_H(\alpha_1 + \alpha_2)$, $AE \leftarrow (R, 2, 1)$;
    for $1 \leq i < j \leq n$ do
      compute $\alpha_i + \alpha_j$;
      if $W_H(\alpha_i) - W_H(\alpha_i + \alpha_j) > weightdecrease$ then
        $weightdecrease \leftarrow W_H(\alpha_i) - W_H(\alpha_i + \alpha_j)$, $AE \leftarrow (R, j, i)$;
      end if
      if $W_H(\alpha_j) - W_H(\alpha_i + \alpha_j) > weightdecrease$ then
        $weightdecrease \leftarrow W_H(\alpha_j) - W_H(\alpha_i + \alpha_j)$, $AE \leftarrow (R, i, j)$;
      end if
    end for
    for $1 \leq i < j \leq n$ do
      compute $\beta_i + \beta_j$;
      if $W_H(\beta_i) - W_H(\beta_i + \beta_j) > weightdecrease$ then
        $weightdecrease \leftarrow W_H(\beta_i) - W_H(\beta_i + \beta_j)$, $AE \leftarrow (C, j, i)$;
      end if
      if $W_H(\beta_j) - W_H(\beta_i + \beta_j) > weightdecrease$ then
        $weightdecrease \leftarrow W_H(\beta_j) - W_H(\beta_i + \beta_j)$, $AE \leftarrow (C, i, j)$;
      end if
    end for
    if $AE = (R, i, j)$ then
      add $\alpha_i$ to $\alpha_j$, output $AE$, $r \leftarrow r + 1$;
    else
      add $\beta_i$ to $\beta_j$, output $AE$, $r \leftarrow r + 1$;
    end if
  end while
  if $W_H(M) > n$ then
    print "Fail.";
  else
    print "Success via $r$ steps.", output $M$;
  end if
```

In Algorithm 3, $r$ is a variable recording the number of additive elementary operations, and $RUB$ is assigned the difference between the Hamming weight of origin matrix $M$

and its order $n$. If $r$ exceeds $RUB$, it is unnecessary to let the program proceed because its output will not be better than computing co-ordinates of output of the linear layer independently. In each while loop, Algorithm 3 looks for an additive elementary operation that can reduce the Hamming weight of $M$ most among all additive row and column elementary operations. If the algorithm finally displays "Success", then we will get a series of additive elementary operations and a permutation matrix. In form of matrix multiplication, we will get $R_r \cdots R_1 M C_1 \cdots C_s = P$, where each $R_i$ and each $C_j$ are additive elementary matrices and $P$ is a permutation matrix. Consequently, we will obtain a P-AE factorization $M = R_1 \cdots R_r P C_s \cdots C_1$.

One important advantage of Algorithm 3 is that its time/space complexity is much lower than that of Algorithm 2. For a linear layer $M \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$, if Algorithm 3 could succeed in $k$ loops, then the time complexity of it is $k(n^2 - n)$. It is a piece of cake in comparison with $(n^2 - n)^k$ – the time complexity of Algorithm 2 with the same number of loops. Therefore, Algorithm 3 is substantially more practical than Algorithm 2 is. Nevertheless, we have to admit that Algorithm 3 does not necessarily give us an optimal implementation of the input linear layer even though it succeeds.

It is well known that implementation costs of the inverses of many linear layers are higher than that of themselves (the linear layer of AES, for example). As we mentioned before, besides using fewer XORs and no extra temporary register, another advantage of MIOSS is implementing every invertible linear layer and its inverse with the same cost. This advantage is also valid to the contents of this section. More specifically, if we get an efficient implementation $M = R_1 \cdots R_r P C_s \cdots C_1$ of a given linear layer $M \in \mathcal{IM}_{n\times n}(\mathbb{F}_2)$ by Algorithm 3, then we immediately obtain an implementation of $M^{-1}$: $M^{-1} = C_1 \cdots C_s P^{-1} R_r \cdots R_1$. Obviously, $M^{-1}$ has the same implementation cost as $M$ does. In fact, we recommend a better strategy – for a given linear layer $M$, one can conduct Algorithm 3 to both $M$ and $M^{-1}$, then choose a better (with a fewer number of additive elementary matrices) one from two implementations as the final option.

### 6.1 A New Efficient Implementation of the Linear Layer of AES

As an application of Algorithm 3, we conduct it to the linear layer of AES and list the results in Appendix B. The linear layer of AES consists of two phases: ShiftRows and MixColumns. We care about nothing of ShiftRows since it is just a permutation to input vectors. MixColumn is the one we really concern about. The Hamming weight of the matrix of encryption MixColumns is 184, and the Hamming weight of the matrix of decryption MixColumns is 440. So, if we implement them by computing co-ordinates of outputs independently, they will cost 152 XORs and 408 XORs, respectively. However, with Algorithm 3, we get an efficient implementation of encryption MixColumns and an efficient implementation of decryption MixColumns. Both of them cost only 120 XORs. Although we cannot guarantee they are optimal implementations, it is still an amazing result.

## 7 Conclusion

In this paper, we first investigate the effect of two implementation strategies on implementation costs of linear layers: computing the co-ordinates of outputs independently and modifying input vectors to outputs step by step. We focus on the latter because it preforms better than the former. Then, we clarify the measurement of implementation cost and optimal implementation procedures of linear layers. Next, to find an optimal implementation procedure of a given linear layer, we construct a graph-theoretical model and transfer the problem to the shortest path problem in graph theory. Then, we adopt a "double-direction" algorithm that uses less storage space and makes the search for a shortest path more efficient in our regular graph. Finally, we construct another algorithm for finding efficient implementations of linear layers. An important advantage of this last algorithm is its extremely low complexity. We conduct it to the linear layer of AES and get very efficient implementations of the linear layer. We hope our work would be beneficial to the design of implementation of linear layers.

**Acknowledgements.** The authors would like to thank Meicheng Liu for sincere discussion and constructive suggestion.

## References

[1] Christof Beierle, Thorsten Kranz, and Gregor Leander. Lightweight multiplication in $GF(2^n)$ with applications to MDS matrices. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 625–653, 2016.

[2] Béla Bollobás. *Modern Graph Theory*, volume 184 of *Graduate Texts in Mathematics*. Springer Science+Business Media, New York City, the U.S., 1998.

[3] Richard A. Brualdi. *Introductory Combinatorics*. Pearson Education, New York City, the U.S., 5th edition, 2009.

[4] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[5] John M. Harris, Jeffry L. Hirst, and Michael J. Mossinghoff. *Combinatorics and Graph Theory*. Undergraduate Texts in Mathematics. Springer Science+Business Media, New York City, the U.S., 2nd edition, 2008.

[6] Kenneth Hoffman. *Linear Algebra*. Prentice-Hall, Englewood Cilffs, the U.S., 2nd edition, 1971.

[7] Yongqiang Li and Mingsheng 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.

[8] Meicheng Liu and Siang Meng 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.

[9] Mahdi Sajadieh, Mohammad Dakhilalian, Hamid Mala, and Pouyan Sepehrdad. Recursive diffusion layers for block ciphers and hash functions. In *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, pages 385–401, 2012.

[10] Sumanta Sarkar and Siang Meng Sim. A deeper understanding of the XOR count distribution in the context of lightweight cryptography. In *Progress in Cryptology - AFRICACRYPT 2016 - 8th International Conference on Cryptology in Africa, Fes, Morocco, April 13-15, 2016, Proceedings*, pages 167–182, 2016.

[11] Siang Meng Sim, Khoongming Khoo, Frédérique E. Oggier, and Thomas Peyrin. Lightweight MDS involution matrices. In *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, pages 471–493, 2015.

[12] Shengbao Wu, Mingsheng Wang, and Wenling Wu. Recursive diffusion layers for (lightweight) block ciphers and hash functions. In *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, pages 355–371, 2012.

## A Experimental Results

In this appendix, every $A_{i,j}$ denotes $I_n + E_{i,j}$ with a proper order $n$.

$$L_1 = \begin{pmatrix} 1&0&1&1&1\\ 1&1&0&1&1\\ 0&1&1&0&1\\ 1&1&1&1&1\\ 1&1&0&0&1 \end{pmatrix} = A_{1,4}A_{4,2}A_{3,4} \begin{pmatrix} 0&1&0&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1\\ 0&0&1&0&0\\ 1&0&0&0&0 \end{pmatrix} A_{4,1}A_{1,5}A_{5,2}.$$

$$L_2 = \begin{pmatrix} 1&1&1&0&1\\ 0&1&1&0&1\\ 0&1&1&1&1\\ 1&1&0&1&1\\ 1&0&1&1&1 \end{pmatrix} = A_{4,1}A_{1,2}A_{5,1}A_{2,3} \begin{pmatrix} 1&0&0&0&0\\ 0&0&0&1&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&1 \end{pmatrix} A_{2,5}A_{5,3}A_{3,5}.$$

$$L_3 = \begin{pmatrix} 1&1&0&0&1\\ 0&1&1&0&1\\ 1&0&1&1&1\\ 1&1&0&1&1\\ 1&1&1&1&1 \end{pmatrix} = A_{3,5}A_{5,3}A_{2,5} \begin{pmatrix} 1&0&0&0&0\\ 0&0&0&0&1\\ 0&1&0&0&0\\ 0&0&0&1&0\\ 0&0&1&0&0 \end{pmatrix} A_{4,1}A_{1,5}A_{5,2}.$$

$$L_4 = \begin{pmatrix} 1&1&0&0&0\\ 1&1&1&1&1\\ 1&1&0&1&0\\ 0&1&1&1&1\\ 1&1&0&1&1 \end{pmatrix} = A_{4,2}A_{2,5}A_{5,3} \begin{pmatrix} 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 1&0&0&0&0\\ 0&0&0&0&1 \end{pmatrix} A_{4,2}A_{2,1}.$$

$$L_5 = \begin{pmatrix} 1&1&1&1&0\\ 1&1&1&0&1\\ 1&0&1&1&0\\ 0&1&1&1&1\\ 0&1&1&1&0 \end{pmatrix} = A_{2,4}A_{4,5}A_{5,1} \begin{pmatrix} 0&1&0&0&0\\ 0&0&0&1&0\\ 0&0&1&0&0\\ 0&0&0&0&1\\ 1&0&0&0&0 \end{pmatrix} A_{2,3}A_{3,4}A_{4,1}.$$

$$L_6 = \begin{pmatrix} 1&0&1&0&1\\ 1&1&0&1&1\\ 1&1&1&1&1\\ 1&1&1&0&0\\ 1&1&1&0&1 \end{pmatrix} = A_{4,5}A_{2,3}A_{3,5} \begin{pmatrix} 1&0&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1\\ 0&1&0&0&0 \end{pmatrix} A_{2,1}A_{1,5}A_{1,3}.$$

$$L_7 = \begin{pmatrix} 0&0&1&1&1\\ 0&1&1&1&0\\ 0&1&1&0&1\\ 0&1&1&1&1\\ 1&1&1&1&1 \end{pmatrix} = A_{3,1}A_{1,4}A_{5,4} \begin{pmatrix} 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1\\ 1&0&0&0&0 \end{pmatrix} A_{5,3}A_{3,4}A_{4,2}.$$

$$L_8 = \begin{pmatrix} 1&1&0&1&1\\ 1&1&1&1&1\\ 1&1&1&1&0\\ 1&1&0&0&1\\ 1&0&0&1&0 \end{pmatrix} = A_{3,2}A_{2,1}A_{1,5} \begin{pmatrix} 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&1\\ 1&0&0&0&0\\ 0&0&0&1&0 \end{pmatrix} A_{1,2}A_{2,5}A_{4,1}.$$

$$L_9 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} = A_{3,4}A_{4,5}A_{5,1} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} A_{3,4}A_{4,5}A_{3,1}.$$

$$L_{10} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} = A_{1,2}A_{2,3}A_{3,5} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} A_{4,2}A_{2,1}A_{3,4}.$$

$$L_{11} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} = A_{3,4}A_{1,3}A_{1,6} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} A_{4,5}A_{1,6}A_{5,2}.$$

$$L_{12} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} = A_{3,1}A_{2,1}A_{6,2} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} A_{6,3}A_{1,5}A_{4,3}.$$

$$L_{13} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} = A_{5,1}A_{4,5}A_{2,4} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} A_{3,4}A_{1,3}A_{5,4}.$$

$$L_{14} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} = A_{1,3}A_{3,5}A_{4,2} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} A_{3,1}A_{6,4}.$$

$$L_{15} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = A_{1,3}A_{2,1}A_{1,6} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} A_{2,4}A_{4,3}A_{1,4}.$$

$$L_{16} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = A_{1,4}A_{2,1}A_{1,6}A_{3,1} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} A_{5,6}A_{5,4}A_{1,2}.$$

**Table 1:** Implementation Cost of Linear Layers

| Linear Layer | $L_1$ | $L_2$ | $L_3$ | $L_4$ | $L_5$ | $L_6$ | $L_7$ | $L_8$ | $L_9$ | $L_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Min-XOR-Count | 6 | 7 | 6 | 5 | 6 | 6 | 6 | 6 | 6 | 6 |
| $W_H(L_i) - n$ | 14 | 14 | 14 | 13 | 13 | 14 | 13 | 13 | 13 | 13 |
| Linear Layer | $L_{11}$ | $L_{12}$ | $L_{13}$ | $L_{14}$ | $L_{15}$ | $L_{16}$ | $L_{17}$ | $L_{18}$ | $L_{19}$ | $L_{20}$ |
| Min-XOR-Count | 6 | 6 | 6 | 5 | 6 | 7 | 6 | 7 | 6 | 6 |
| $W_H(L_i) - n$ | 8 | 7 | 8 | 7 | 7 | 8 | 8 | 8 | 8 | 8 |

$$L_{17} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} = A_{1,2}A_{2,4}A_{6,1} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} A_{2,4}A_{3,2}A_{6,2}.$$

$$L_{18} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} = A_{4,1}A_{1,2}A_{6,1}A_{3,6} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} A_{6,1}A_{5,2}A_{1,3}.$$

$$L_{19} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} = A_{5,4}A_{5,2}A_{1,6} \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} A_{3,2}A_{1,4}A_{6,3}.$$

$$L_{20} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} = A_{4,6}A_{3,4}A_{3,2} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} A_{1,4}A_{1,2}A_{4,3}.$$

## B An Efficient Implementation of Linear Layers of AES

Let $MC$ denote the matrix of encryption MixColumns of AES. Then

$$MC = \begin{pmatrix} M(0x02) & M(Ox03) & M(Ox01) & M(Ox01) \\ M(0x01) & M(0x02) & M(0x03) & M(0x01) \\ M(0x01) & M(0x01) & M(0x02) & M(0x03) \\ M(0x03) & M(0x01) & M(0x01) & M(0x02) \end{pmatrix} \in \mathcal{M}_{32\times 32}(\mathbb{F}_2),$$

where

$$M(0x02) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, M(Ox03) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$M(0x01)=\begin{pmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{pmatrix}.$$

Meanwhile, the matrix of decryption MixColumns of AES

$$MC^{-1}=\begin{pmatrix}M(0x0e)&M(0x0b)&M(0x0d)&M(0x09)\\M(0x09)&M(0x0e)&M(0x0b)&M(0x0d)\\M(0x0d)&M(0x09)&M(0x0e)&M(0x0b)\\M(0x0b)&M(0x0d)&M(0x09)&M(0x0e)\end{pmatrix}\in\mathcal{M}_{32\times32}(\mathbb{F}_2),$$

where

$$M(0x0e)=\begin{pmatrix}0&1&1&1&0&0&0&0\\1&0&1&1&1&0&0&0\\0&1&0&1&1&1&0&0\\0&0&1&0&1&1&1&0\\0&1&1&0&0&1&1&1\\0&1&0&0&0&0&1&1\\0&0&1&0&0&0&0&1\\1&1&1&0&0&0&0&0\end{pmatrix},M(0x0b)=\begin{pmatrix}1&1&0&1&0&0&0&0\\1&1&1&0&1&0&0&0\\0&1&1&1&0&1&0&0\\0&0&1&1&1&0&1&0\\0&1&1&0&1&1&0&1\\0&1&0&0&0&1&1&0\\0&0&1&0&0&0&1&1\\1&1&1&0&0&0&0&1\end{pmatrix},$$

$$M(0x0d)=\begin{pmatrix}1&0&1&1&0&0&0&0\\1&1&0&1&1&0&0&0\\0&1&1&0&1&1&0&0\\0&0&1&1&0&1&1&0\\0&1&1&0&1&0&1&1\\0&1&0&0&0&1&0&1\\0&0&1&0&0&0&1&0\\1&1&1&0&0&0&0&1\end{pmatrix},M(0x09)=\begin{pmatrix}1&0&0&1&0&0&0&0\\1&1&0&0&1&0&0&0\\0&1&1&0&0&1&0&0\\0&0&1&1&0&0&1&0\\0&1&1&0&1&0&0&1\\0&1&0&0&0&1&0&0\\0&0&1&0&0&0&1&0\\1&1&1&0&0&0&0&1\end{pmatrix}.$$

We have $W_H(MC)=184$ and $W_H(MC^{-1})=440$.

We conduct Algorithm 3 to $MC$ but get nothing useful. However, when we conduct Algorithm 3 to $MC^{-1}$ and set $b=-1$, we get a P-AE factorization of $MC^{-1}$ with 120 additive elementary matrices:

$$\begin{aligned}MC^{-1}=\;&A_{18,2}A_{10,26}A_{25,9}A_{19,3}A_{5,21}A_{13,29}A_{21,29}A_{20,21}A_{17,1}A_{1,19}A_{8,16}A_{16,24}A_{24,32}\\&A_{30,14}A_{29,7}A_{6,22}A_{2,26}A_{9,2}A_{2,19}A_{10,2}A_{3,5}A_{11,13}A_{27,13}A_{7,15}A_{22,7}A_{19,20}\\&A_{2,18}A_{4,6}A_{26,4}A_{21,6}A_{32,8}A_{9,17}A_{12,30}A_{15,16}A_{32,17}A_{26,21}A_{22,30}A_{29,22}A_{28,30}\\&A_{4,28}A_{1,10}A_{2,10}A_{3,2}A_{11,2}A_{14,2}A_{17,2}A_{22,2}A_{25,2}A_{1,25}A_{27,2}A_{12,4}A_{12,21}\\&A_{19,12}A_{13,4}A_{5,13}A_{29,5}A_{18,10}A_{26,11}A_{26,17}A_{26,19}A_{27,19}A_{26,25}A_{32,25}A_{13,6}\\&A_{13,30}A_{20,13}A_{11,20}A_{27,13}A_{11,27}A_{12,11}A_{32,12}A_{7,32}A_{31,7}A_{23,31}A_{23,32}A_{6,23}\\&A_{30,23}\boldsymbol{P}A_{32,24}A_{30,22}A_{16,24}A_{30,16}A_{15,31}A_{15,27}A_{30,15}A_{13,30}A_{13,24}A_{29,13}A_{21,29}\\&A_{18,12}A_{8,16}A_{5,28}A_{5,21}A_{4,29}A_{4,20}A_{4,5}A_{3,4}A_{18,3}A_{2,27}A_{2,18}A_{13,28}A_{23,27}\\&A_{30,23}A_{7,8}A_{4,12}A_{28,4}A_{23,32}A_{31,7}A_{31,23}A_{7,15}A_{12,27}A_{1,17}A_{22,6}A_{14,30}A_{9,25}\\&A_{27,19}A_{26,18}A_{27,11}A_{18,10}A_{19,3}A_{10,2},\end{aligned}$$

where each $A_{i,j}=I_{32}+E_{i,j}$ and $P$ is a permutation matrix described in Table 2.

**Table 2:** The Permutation $P$ in the P-AE Factorization of $MC^{-1}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\rho(i)$ | 17 | 26 | 3 | 29 | 5 | 22 | 7 | 8 | 25 | 10 | 20 | 27 | 28 | 30 | 15 | 16 |
| $i$ | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| $\rho(i)$ | 1 | 2 | 19 | 12 | 21 | 6 | 31 | 32 | 9 | 18 | 11 | 4 | 13 | 14 | 23 | 24 |

- $\rho(i)$ is the column index of nonzero entry in $i$-th row of $P$.

Consequently, we get a P-AE factorization of $MC$ with 120 additive elementary matrices too.

$$\begin{aligned} MC = \; & (MC^{-1})^{-1} \\ = \; & A_{10,2}A_{19,3}A_{18,10}A_{27,11}A_{26,18}A_{27,19}A_{9,25}A_{14,30}A_{22,6}A_{1,17}A_{12,27}A_{7,15}A_{31,23} \\ & A_{31,7}A_{23,32}A_{28,4}A_{4,12}A_{7,8}A_{30,23}A_{23,27}A_{13,28}A_{2,18}A_{2,27}A_{18,3}A_{3,4}A_{4,5}A_{4,20} \\ & A_{4,29}A_{5,21}A_{5,28}A_{8,16}A_{18,12}A_{21,29}A_{29,13}A_{13,24}A_{13,30}A_{30,15}A_{15,27}A_{15,31}A_{30,16} \\ & A_{16,24}A_{30,22}A_{32,24}\boldsymbol{P^{-1}}A_{30,23}A_{6,23}A_{23,32}A_{23,31}A_{31,7}A_{7,32}A_{32,12}A_{12,11}A_{11,27} \\ & A_{27,13}A_{11,20}A_{20,13}A_{13,30}A_{13,6}A_{32,25}A_{26,25}A_{27,19}A_{26,19}A_{26,17}A_{26,11}A_{18,10} \\ & A_{29,5}A_{5,13}A_{13,4}A_{19,12}A_{12,21}A_{12,4}A_{27,2}A_{1,25}A_{25,2}A_{22,2}A_{17,2}A_{14,2}A_{11,2}A_{3,2} \\ & A_{2,10}A_{1,10}A_{4,28}A_{28,30}A_{29,22}A_{22,30}A_{26,21}A_{32,17}A_{15,16}A_{12,30}A_{9,17}A_{32,8}A_{21,6} \\ & A_{26,4}A_{4,6}A_{2,18}A_{19,20}A_{22,7}A_{7,15}A_{27,13}A_{11,13}A_{3,5}A_{10,2}A_{2,19}A_{9,2}A_{2,26}A_{6,22} \\ & A_{29,7}A_{30,14}A_{24,32}A_{16,24}A_{8,16}A_{1,19}A_{17,1}A_{20,21}A_{21,29}A_{13,29}A_{5,21}A_{19,3}A_{25,9} \\ & A_{10,26}A_{18,2}. \end{aligned}$$