CHAPTER 6

DEVELOPMENT OF LAO-3D LIGHTWEIGHT BLOCK CIPHER

6.1 Introduction

This chapter highlights the design structure of the 3D cipher that has been adopted. In order to develop a new lightweight algorithm based on the 3D cipher, the formulation of a 3D rotation function is presented in Section 6.2. The 3D rotation is the main component in the development of the lightweight block cipher as discussed in Section 6.3. Design construction of the new lightweight block cipher is explained in detail to provide a clear overview of the algorithm developed for security products implementation.

6.2 Formulation of 3D Rotation Function

This section presents the design of a secure cryptographic component in order to fulfil *Research Objective 2*, in which a formulation of the 3D rotation method is developed. The 3D rotation method is suitable to be implemented in any block cipher with a block size that can be the square root of three (e.g., 64, 216, 512, and 1,000). Thus, the 3D design does not increase the block and key sizes of the block cipher algorithm. 64 bits block cipher is the ideal size of an algorithm for 3D design adoption to fit in the implementation of security products in mobile phones which requires a compact encryption algorithm with small computing power to optimize the utilization of the device resources (Rahim et al., 2018).

The selection of the 3D cipher design approach in this research is due to its capability to enhance the security strength of block cipher with improvements of the confusion and diffusion properties. Enhancement made to the new 3D cipher mainly aims to increase security without enlarging the size of the block cipher algorithm, which improves existing 3D cipher methods.

6.2.1 3D Bit Rotation

As the foundation of the development of the 3D rotation component in Section 6.2.2, a *3DBitRotation* function is introduced to perform the permutation operation on the 3D (4 x 4 x 4 bits) cipher state as shown in Figure 6.1. Every portion of the cipher state consists of 16 bits plaintext that is assigned as the *Slice*. In total, four *Slices* are required to accommodate all 64 bits block size of the new lightweight block cipher that is discussed in Section 6.3.

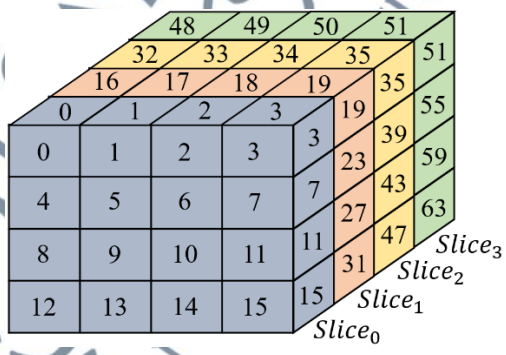


Figure 6.1: 3-Dimensional Cipher State

The *3DBitRotation* function can be applied on the *X*-axis, *Y*-axis, or *Z*-axis. Figure 6.2 shows the rotation formation of the *3DBitRotation* function that is rotated in a specific clockwise direction as follows:

i) $Slice_0$: Rotate 0^o (no rotation).

ii) $Slice_1$: Rotate 90^o.

iii) $Slice_2$: Rotate 180^o.

iv) $Slice_3$: Rotate 270^o.

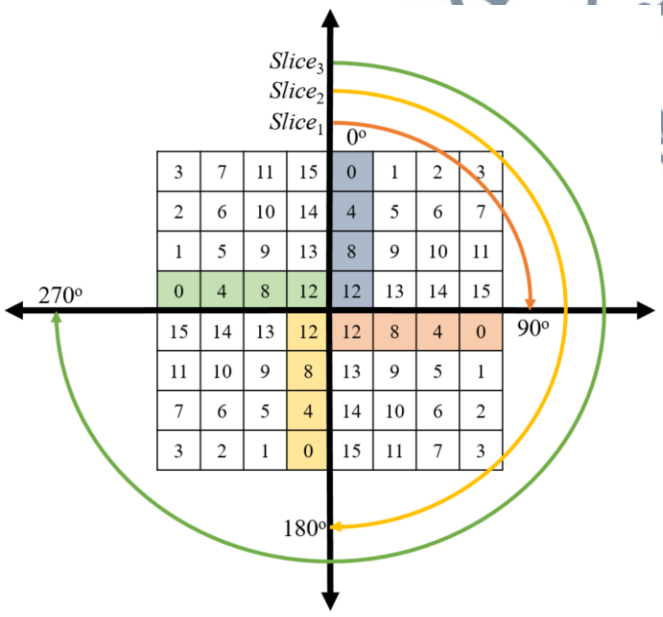


Figure 6.2: 3D Bit Rotation Formation

6.2.2 3D Rotation Function

A new 3D rotation function is designed called *Double3DRotation* that combines three sub-functions including the *3DBitRotation_X-axis*, *AddRoundKey*, and *3DBitRotation_Z-axis* to answer *Research Question 4*. In this rotation function, a plaintext and key are the input of the 3D cipher design to execute the encryption process.

The overall structure of the *Double3DRotation* function is displayed in Figure 6.3. This component can be easily plugged into any block cipher algorithm where the block size is divisible by n^3 (e.g., 64-bit and 512-bit). Details of every function of the *Double3DRotation* are specified in the following subsections.

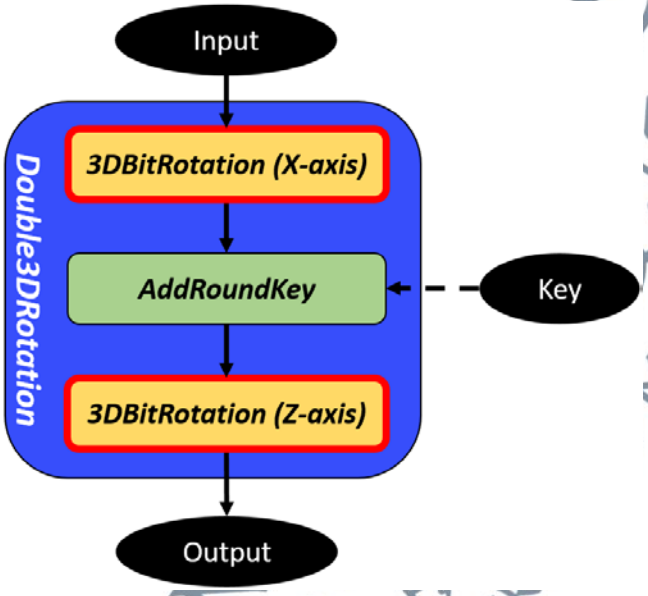


Figure 6.3: 3D Rotation Process

6.2.2.1 3D Bit Rotation (*X*-axis) Function

Implementation of 3D cipher requires the input data string to be transformed into a 3D state which consists of four data *Slices* by placing the data bits into 4 x 4 x 4 matrices. The input data of the cipher or plaintext is represented by $W = w_{63}\|\ldots\|w_1\|w_0$ as displayed in Figure 6.4.

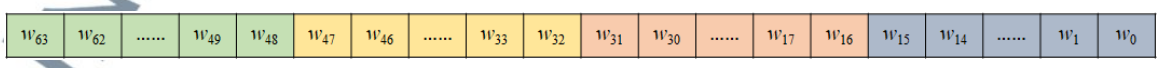


Figure 6.4: 2D Plaintext

The initial 16 bits plaintext, $w_{15}\|\ldots\|w_1\|w_0$ are ordered in $Slice_0$ defined as $a_{0,0}\|\ldots\|a_{0,14}\|a_{0,15}$ and the subsequent 16 bits $w_{31}\|\ldots\|w_{17}\|w_{16}$ are located in $Slice_1$ defined as $a_{1,0}\|\ldots\|a_{1,14}\|a_{1,15}$. The consecutive 16 bits of the plaintext are assigned as $Slice_2$ defined as $a_{2,0}\|\ldots\|a_{2,14}\|a_{2,15}$ and $Slice_3$ defined as $a_{3,0}\|\ldots\|a_{3,14}\|a_{3,15}$ which completed the 3D cipher state as presented in Figure 6.5.

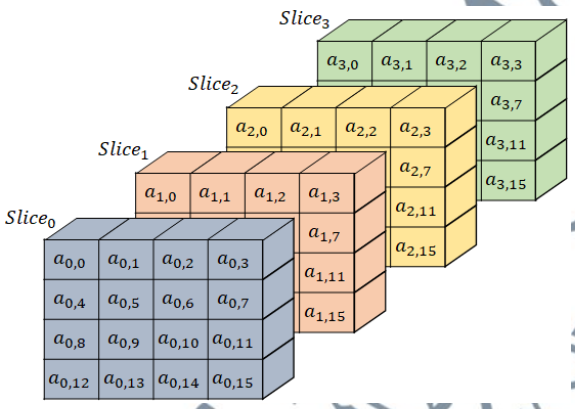


Figure 6.5: 3D Cipher State

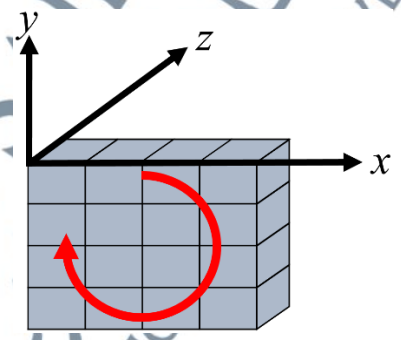


Figure 6.6: *X*-axis Rotation

The 3D cipher state is then rotated in a specific clockwise direction at the *X*-axis as shown in Figure 6.6 with 90^{o} rotation for $Slice_1$, 180^{o} rotation for $Slice_2$, 270^{o} rotation for $Slice_3$, and no rotation for $Slice_0$. The cipher state of each *Slice* before and after operating the *3DBitRotation_X-axis* function is depicted in Figure 6.7.

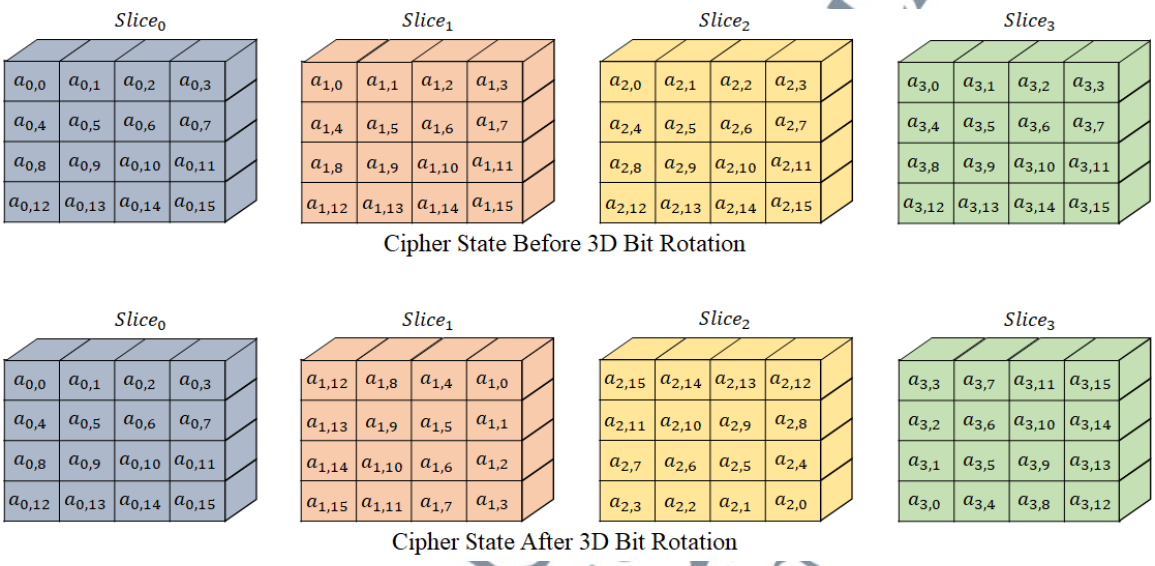


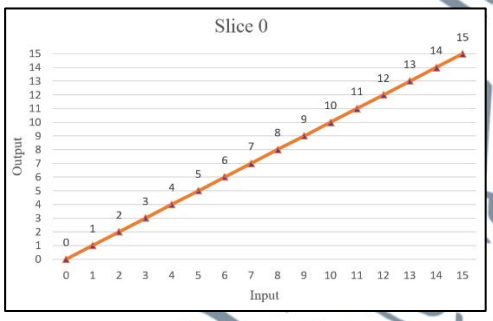
Figure 6.7: 3D Bit Rotation (*X*-axis) Cipher State

Formulation of the *3DBitRotation_X-axis* function can be represented as linear equations as listed in Table 6.1 that are derived from the 3D Bit Rotation Formation shown in Figure 6.2. The *3DBitRotation_X-axis* formulation is divided into four *Slices* that consist of 16 bits output that is generated from the equations.

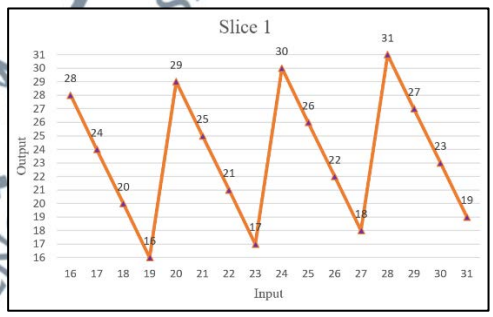
Table 6.1: Formulation of 3D Bit Rotation (*X*-axis)

Slice	Input, *X*	Equation	Output, *Y*
$Slice_0$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15	$Y = X$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
$Slice_1$	16, 17, 18, 19	$Y = -4X + 92$	28, 24, 20, 16
	20, 21, 22, 23	$Y = -4X + 109$	29, 25, 21, 17
	24, 25, 26, 27	$Y = -4X + 126$	30, 26, 22, 18
	$28 \leq X \leq 31$	$Y = -4X + 143$	28, 29, 30, 31
$Slice_2$	32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47	$Y = -X + 79$	47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32
$Slice_3$	48, 49, 50, 51	$Y = 4X - 141$	51, 55, 59, 63
	52, 53, 54, 55	$Y = 4X - 158$	50, 54, 58, 62
	56, 57, 58, 59	$Y = 4X - 175$	49, 53, 57, 61
	60, 61, 62, 63	$Y = 4X - 192$	48, 52, 56, 60

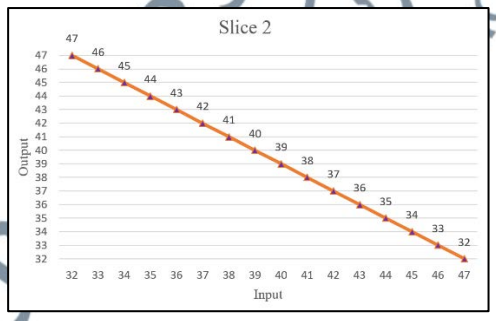
There are ten independent equations derived from the *3DBitRotation_X-axis* function that is dedicated to each *Slice* and its input data. More than one linear equation produced from the function shows that no *Slices* are correlated with one another. For better observation, the graphical outputs of the *3DBitRotation_X-axis* are displayed in Figure 6.8.



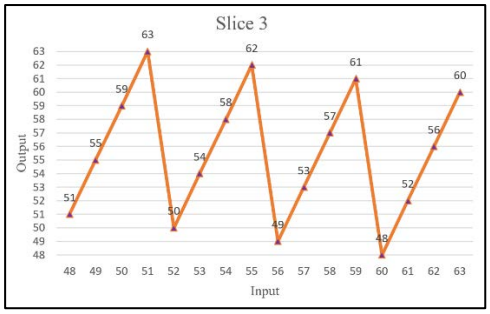
(i) $Slice_0$



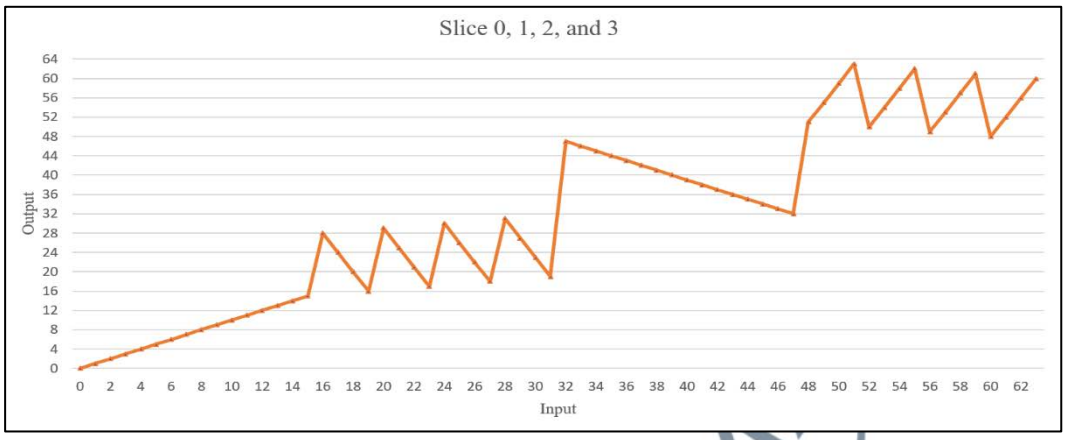
(ii) $Slice_1$



(iii) $Slice_2$



(iv) $Slice_3$



(v) Combination of $Slice_0$, $Slice_1$, $Slice_2$, and $Slice_3$

Figure 6.8: 3D Bit Rotation (*X*-axis) Output

The presented graphs indicate that the distributions of output generated from the *3DBitRotation_X-axis* function are different from each *Slice*, thus no correlation exists. 3D cipher is able to withstand cryptanalytic attacks and statistical analyses, hence, making a cryptographic algorithm more secure (Sahasrabuddhe & Laiphrakpam, 2021). Therefore, the enhanced 3D rotation function can offer better diffusion property to the new block cipher algorithm.

6.2.2.2 Add Round Key Function

AddRoundKey is an XOR operation of the round key (k_0 to k_{63}) and the present state ($Slice_0$ to $Slice_3$) which are combined to form a data string as presented in Figure 6.9. The purpose of implementing the *AddRoundKey* function is to add confusion property in a block cipher algorithm. This function is repeatedly used depending on the number of round keys generated by the key schedule algorithm of a block cipher.

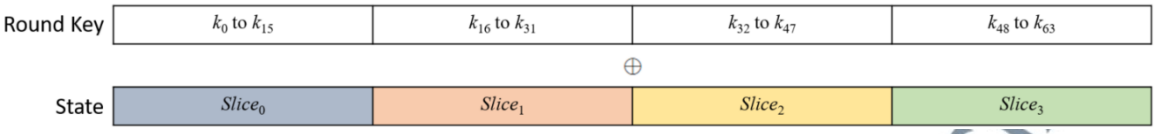


Figure 6.9: Add Round Key

The specific state used in the *AddRoundKey* operation for the new 3D cipher design is the output of the previous *3DBitRotation_X-axis* function. Figure 6.10 shows the state of every *Slice* that contains different data values that are updated after each encryption round with the addition of round keys.

k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}
$\oplus$															
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$a_{0,6}$	$a_{0,7}$	$a_{0,8}$	$a_{0,9}$	$a_{0,10}$	$a_{0,11}$	$a_{0,12}$	$a_{0,13}$	$a_{0,14}$	$a_{0,15}$
=															
$a'_{0,0}$	$a'_{0,1}$	$a'_{0,2}$	$a'_{0,3}$	$a'_{0,4}$	$a'_{0,5}$	$a'_{0,6}$	$a'_{0,7}$	$a'_{0,8}$	$a'_{0,9}$	$a'_{0,10}$	$a'_{0,11}$	$a'_{0,12}$	$a'_{0,13}$	$a'_{0,14}$	$a_{0,15}$

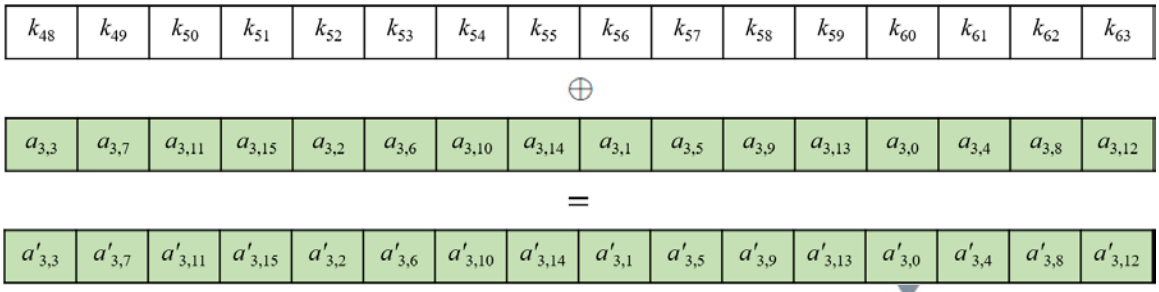
(i) Add Round Key $Slice_0$

k_{16}	k_{17}	k_{18}	k_{19}	k_{20}	k_{21}	k_{22}	k_{23}	k_{24}	k_{25}	k_{26}	k_{27}	k_{28}	k_{29}	k_{30}	k_{31}
$\oplus$															
$a_{1,12}$	$a_{1,8}$	$a_{1,4}$	$a_{1,0}$	$a_{1,13}$	$a_{1,9}$	$a_{1,5}$	$a_{1,1}$	$a_{1,14}$	$a_{1,10}$	$a_{1,6}$	$a_{1,2}$	$a_{1,15}$	$a_{1,11}$	$a_{1,7}$	$a_{1,3}$
=															
$a'_{1,12}$	$a'_{1,8}$	$a'_{1,4}$	$a'_{1,0}$	$a'_{1,13}$	$a'_{1,9}$	$a'_{1,5}$	$a'_{1,1}$	$a'_{1,14}$	$a'_{1,10}$	$a'_{1,6}$	$a'_{1,2}$	$a'_{1,15}$	$a'_{1,11}$	$a'_{1,7}$	$a'_{1,3}$

(ii) Add Round Key $Slice_1$

k_{32}	k_{33}	k_{34}	k_{35}	k_{36}	k_{37}	k_{38}	k_{39}	k_{40}	k_{41}	k_{42}	k_{43}	k_{44}	k_{45}	k_{46}	k_{47}
$\oplus$															
$a_{2,15}$	$a_{2,14}$	$a_{2,13}$	$a_{2,12}$	$a_{2,11}$	$a_{2,10}$	$a_{2,9}$	$a_{2,8}$	$a_{2,7}$	$a_{2,6}$	$a_{2,5}$	$a_{2,4}$	$a_{2,3}$	$a_{2,2}$	$a_{2,1}$	$a_{2,0}$
=															
$a'_{2,15}$	$a'_{2,14}$	$a'_{2,13}$	$a'_{2,12}$	$a'_{2,11}$	$a'_{2,10}$	$a'_{2,9}$	$a'_{2,8}$	$a'_{2,7}$	$a'_{2,6}$	$a'_{2,5}$	$a'_{2,4}$	$a'_{2,3}$	$a'_{2,2}$	$a'_{2,1}$	$a'_{2,0}$

(iii) Add Round Key $Slice_2$



(iv) Add Round Key *Slice*$_3$

Figure 6.10: Add Round Key (*Slice*$_0$ to *Slice*$_3$)

6.2.2.3 3D Bit Rotation (*Z*-axis) Function

This function applies the same method as in the previous *3DBitRotation_X-axis* function. However, the 3D cipher state is rotated at the Z-axis as shown in Figure 6.11 where the rotation direction remains the same as the previous function with 90^o rotation for *Slice*$_1$, 180^o rotation for *Slice*$_2$, 270^o rotation for *Slice*$_3$, and no rotation for *Slice*$_0$. The input of this function is the output from the *AddRoundKey* as shown in Figure 6.12.

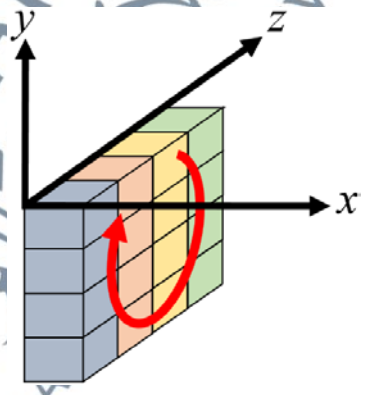


Figure 6.11: *Z*-axis Rotation

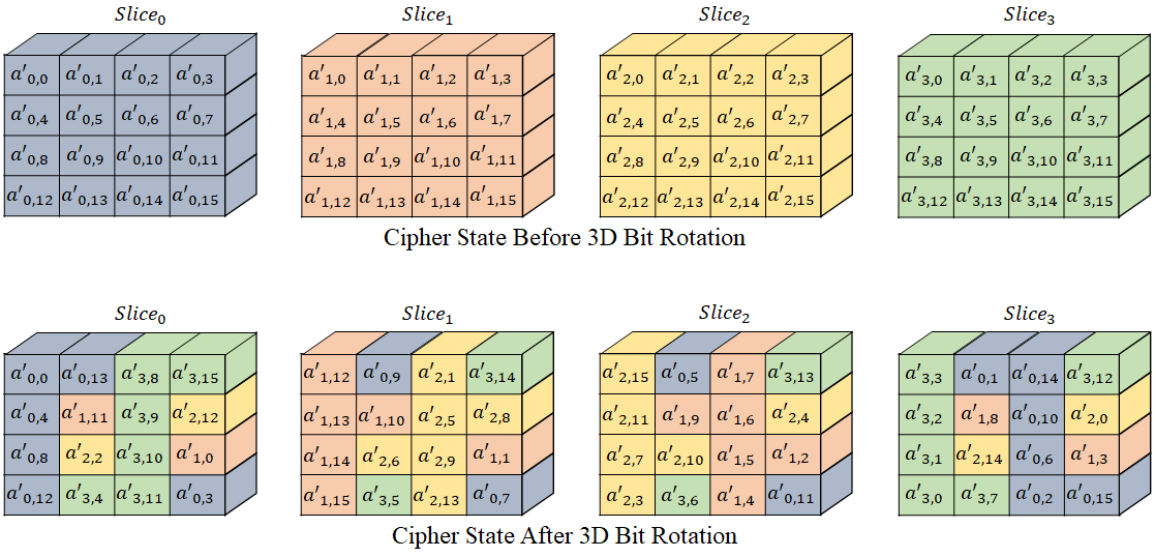


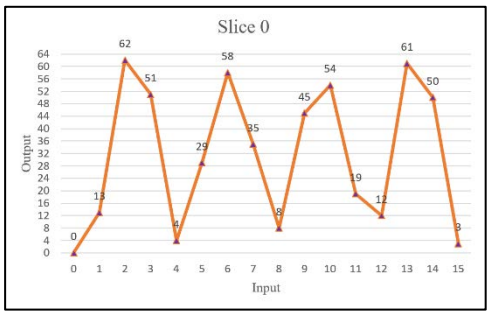
Figure 6.12: 3D Bit Rotation (*Z*-axis) Cipher State

Formulation of the *3DBitRotation_Z-axis* function is derived from the 3D Bit Rotation Formation that is represented in linear equations as listed in Table 6.2. The *3DBitRotation_Z-axis* formulation is divided into four *Slices* that contain 16 bits outputs that are produced from the linear equations.

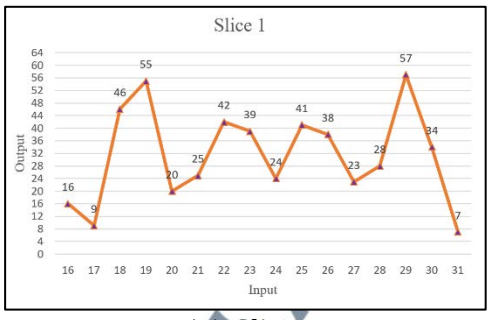
Table 6.2: Formulation of 3D Bit Rotation (*Z*-axis)

Slice	Input, *X*	Equation	Output, *Y*
$Slice_0$	0, 1, 15	$Y = 13X$	0, 13, 3
	2, 3, 5	$Y = -11X + 84$	62, 51, 29
	4, 8, 12	$Y = 49X - 576$	4, 8, 12
	6, 7	$Y = -23X + 196$	58, 35
	9, 10	$Y = 9X - 36$	45, 54
	11, 13, 14	$Y = -11X + 204$	19, 61, 50
$Slice_1$	16, 17, 24	$Y = -7X + 128$	16, 9, 24
	18, 23	$Y = 37X - 620$	46, 39
	19, 21, 22	$Y = 17X - 332$	55, 25, 42
	20, 28	$Y = -15X + 384$	20, 28
	26, 27, 29	$Y = -15X + 428$	38, 23, 57
	25, 30, 31	$Y = -27X + 844$	41, 34, 7
$Slice_2$	32, 36, 37	$Y = -15X + 576$	32, 36, 21
	33, 38, 39	$Y = 17X - 620$	5, 26, 43
	34, 35	$Y = 29X - 956$	30, 59
	40, 43, 44	$Y = 17X - 704$	40, 27, 44
	41, 42, 47	$Y = -15X + 652$	37, 22, 11
	45, 46	$Y = -35X + 1628$	53, 18
$Slice_3$	48, 49, 52, 56	$Y = -47X + 2304$	48, 1, 52, 56
	50, 51	$Y = 49X - 2436$	14, 63
	53	$Y = -7X + 388$	17
	54, 55	$Y = 37X - 1988$	10, 47
	57, 58	$Y = -27X + 1572$	33, 6
	59	$Y = 25X - 1444$	31
	60, 61	$Y = -11X + 720$	60, 49
	62, 63	$Y = 13X - 804$	2, 15

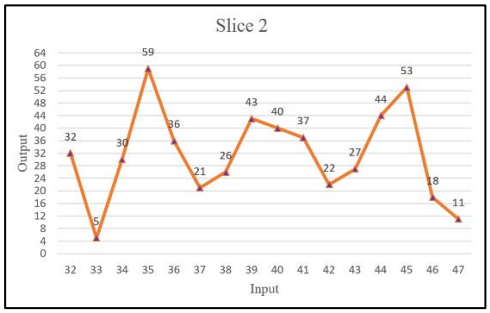
In total, 26 independent equations were derived from the *3DBitRotation_Z-axis* function that is pointed to a specific *Slice* and its input data. Multiple linear equations produced from the function indicate that the *Slices* are not correlated with one another. Graphical representations of the output from the *3DBitRotation_Z-axis* function are displayed in Figure 6.13.



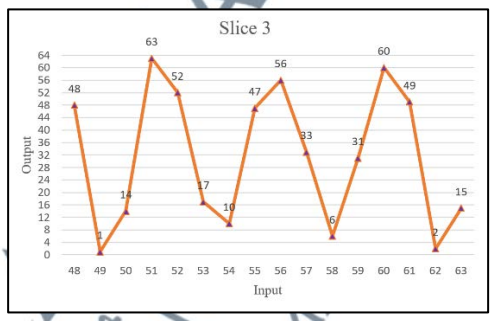
(i) *Slice*$_0$



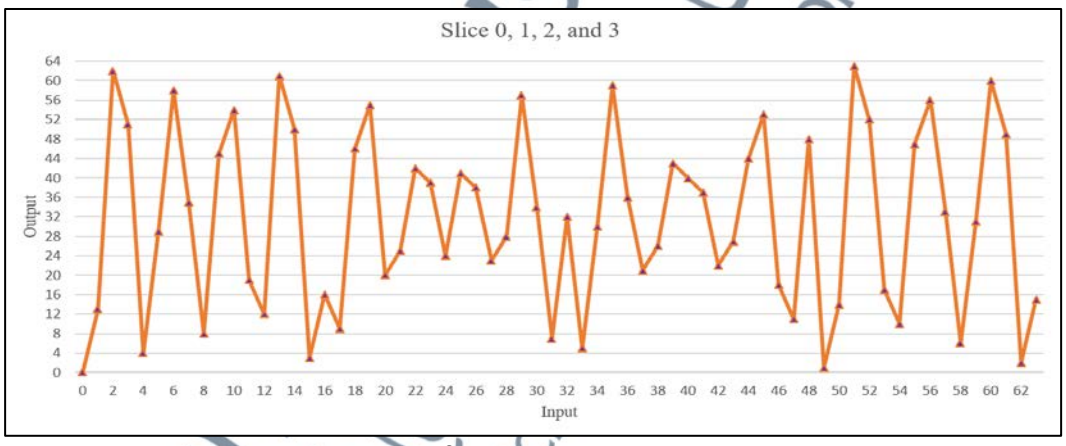
(ii) *Slice*$_1$



(iii) *Slice*$_2$



(iv) *Slice*$_3$



(v) Combination of *Slice*$_0$, *Slice*$_1$, *Slice*$_2$, and *Slice*$_3$

Figure 6.13: 3D Bit Rotation (*Z*-axis) Output

The graphs show that the distributions of output produced from the *3DBitRotation_Z-axis* function are not identical, thus no correlation exists between each *Slice*. Therefore, the rotation function can provide significant diffusion property to the new lightweight block cipher.

6.3 Development of LAO-3D Lightweight Block Cipher

This section highlights the development of the new lightweight block cipher algorithm called Light Algorithm Operation (LAO-3D) based on 3-dimensional (3D) permutation to answer *Research Question 5* and to fulfil *Research Objective 3*, thus producing *Research Contribution 2*. LAO-3D is specifically designed for security products as it is more compact and requires small computing power. The algorithm provides security at a low cost due to its design simplicity which is developed based on the highly efficient RECTANGLE block cipher. In order to improve the security strength of the original cipher design, the new algorithm implemented the 3D rotation method introduced in Section 6.2. The combination of the 3D rotation method and improved RECTANGLE structure has resulted in the development of a secure LAO-3D lightweight block cipher.

In the following subsections, the general specifications of the new algorithm are discussed to introduce an overview of its design structure. Next, the encryption algorithm is presented by detailing each of the cryptographic functions used in the lightweight block cipher. This is followed by the introduction of the key schedule algorithm along with its design components.

6.3.1 Algorithm Specifications

Light Algorithm Operation (LAO-3D) based on 3-dimensional permutation is a lightweight block cipher with 20 iterated encryption rounds. LAO-3D is built with 64-bit block size and 128-bit key size. The cipher state is divided into four rows for its cipher operations which implements the Substitution-Permutation Network (SPN) structure. SPN structure is adopted as it takes lesser execution time, consumes lower energy

implementation, and requires a smaller number of rounds function compared to the Feistel network (Biswas et al., 2020).

Every encryption round of LAO-3D cipher consists of three main features. First, the linear layer guarantees high diffusion over multiple encryption rounds. Second, the non-linear layer has a parallel application of the S-box that has high nonlinearity properties. Third, the key addition layer applies a simple XOR of the round key to the intermediate cipher state that introduced confusion property to the algorithm. Implementation of the linear and non-linear layers in LAO-3D would provide diffusion and confusion properties in the algorithm which are the fundamental requirements of a secure lightweight block cipher (Pehlivanoğlu et al., 2017).

The different transformations operated on the intermediate result of this cipher is called the cipher state. The cipher state can be pictured as a rectangular array of bits. This array has four rows while the number of columns is equal to the block size (64 bits) divided by the number of rows. On the other hand, the secret key is similarly pictured as a rectangular array with four rows. The number of columns of the secret key is equal to the key size (128 bits) divided by four rows. These representations of the cipher state and key state are illustrated in Figure 6.14.

$a_{0,15}$	$a_{0,14}$	...	$a_{0,2}$	$a_{0,1}$	$a_{0,0}$
$a_{1,15}$	$a_{1,14}$	...	$a_{1,2}$	$a_{1,1}$	$a_{1,0}$
$a_{2,15}$	$a_{2,14}$	...	$a_{2,2}$	$a_{2,1}$	$a_{2,0}$
$a_{3,15}$	$a_{3,14}$	...	$a_{3,2}$	$a_{3,1}$	$a_{3,0}$

Cipher State

v_{31}	v_{30}	...	v_2	v_1	v_0
v_{63}	v_{62}	...	v_{34}	v_{33}	v_{32}
v_{95}	v_{94}	...	v_{66}	v_{65}	v_{64}
v_{127}	v_{126}	...	v_{98}	v_{97}	v_{96}

Key State

Figure 6.14: Cipher State and Key State

6.3.2 Encryption Algorithm

The round transformation of LAO-3D block cipher is composed of four different functions as shown in the following pseudo C notation:

RoundKeysGeneration (*Key*)
AddRoundKey (*CipherState*, *RoundKey*$_0$)
for i = 1 *to* 20
{
    SubColumn (*CipherState*);
    Double3DRotation (*CipherState*, *RoundKey*$_i$);
}

In this notation, the *AddRoundKey*, *SubColumn*, and *Double3DRotation* functions are operated on arrays in which the inputs are the *State* and *RoundKey*. The overall encryption and decryption process of LAO-3D block cipher is displayed in Figure 6.15.

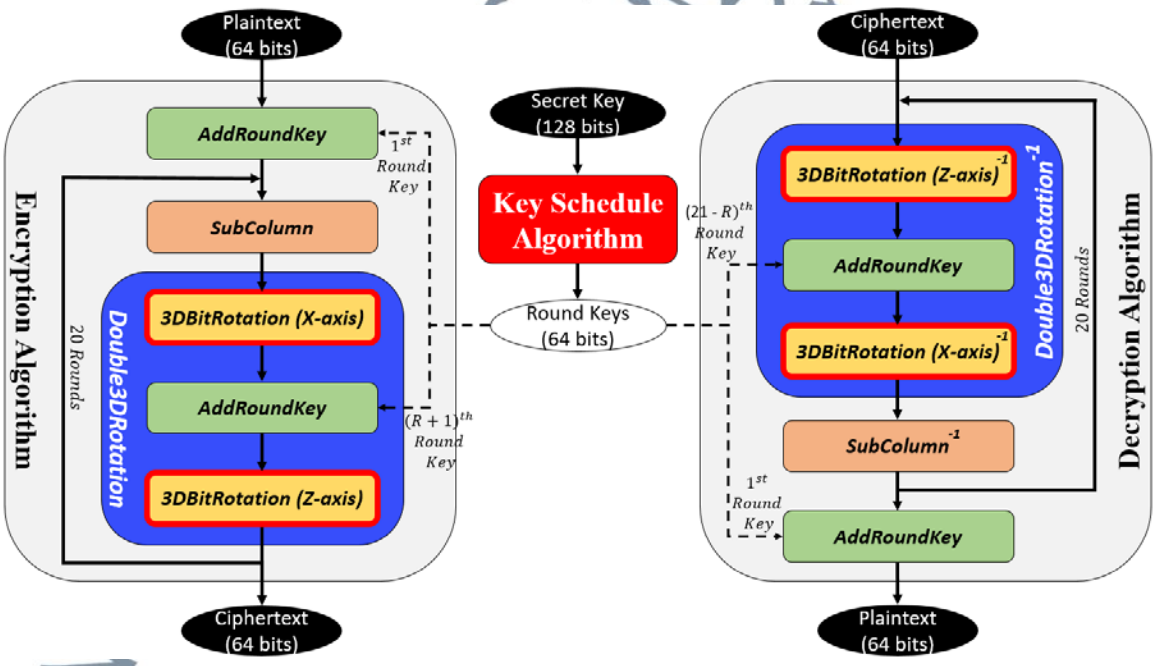


Figure 6.15: LAO-3D Encryption and Decryption Process

For the decryption process, the functions are implemented in reverse order starting with rounds of the *Double3DRotation* and *SubColumn* functions, then followed by the *AddRoundKey*. The round keys are used in reverse order for the decryption process. Every functions transformation of the algorithm is shown in the following pseudo C notation:

```
RoundKeysGeneration (Key)
for i = 1 to 20
{
    Double3DRotation-1 (CipherState, RoundKey21-i);
    SubColumn-1 (CipherState);
}
AddRoundKey (CipherState, RoundKey0)
```

6.3.2.1 Add Round Key

In the *AddRoundKey* operation, the plaintext defined as $A = a_{3,15} \dots a_{3,0} \| a_{2,15} \dots a_{2,0} \| a_{1,15} \dots a_{1,0} \| a_{0,15} \dots a_{0,0}$ applied a simple bitwise XOR with the round key defined as $K = k_{3,15} \dots k_{3,0} \| k_{2,15} \dots k_{2,0} \| k_{1,15} \dots k a_{1,0} \| k_{0,15} \dots k_{0,0}$ to produce the cipher state as shown in Figure 6.16. The 64 bits round key is generated using the key schedule algorithm as described in Section 6.3.3. For this function, the input and output of *AddRoundKey* used in the encryption and decryption processes are displayed in Table 6.3.

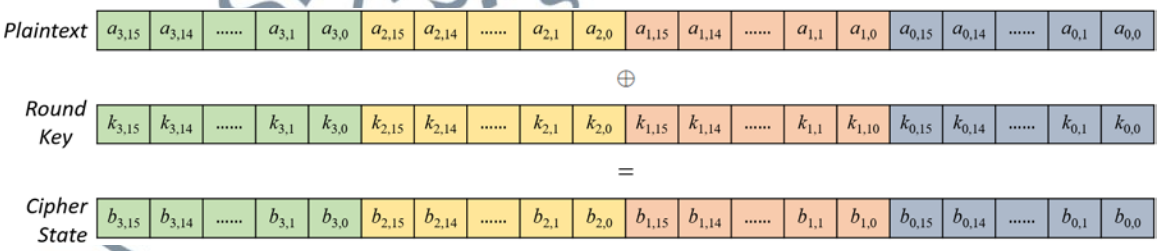


Figure 6.16: Add Round Key

Table 6.3: Input and Output of Add Round Key

	Input	Output
Encryption	Plaintext	Input of *SubColumn*
Decryption	Output of *SubColumn*	Plaintext

6.3.2.2 Sub Column

SubColumn is a non-linear bit substitution that operates independently on each column (4 bits) of the state. This function implemented the substitution box using the PRESENT S-box (Bogdanov et al., 2007) as shown in **Table 6.4**. The PRESENT S-box is implemented to replace the original RECTANGLE S-box due to its satisfactory cryptographic properties as well as its small hardware footprint. There are many lightweight algorithms implemented PRESENT S-box such as PriPresent (Girija et al., 2020), µ2 (Yeoh et al., 2020), LED (Guo et al., 2011), PHOTON (Guo et al., 2011), and GOST (Poschmann et al., 2010). This S-box function is a four by four bits substitution box where the output distribution of the S-box can be observed in Figure 6.17.

Table 6.4: S-box

Hexadecimal		Decimal		Binary	
Input, *x*	**Output, *S*(*x*)**	**Input, *x***	**Output, *S*(*x*)**	**Input, *x***	**Output, *S*(*x*)**
0	C	0	12	0000	1100
1	5	1	5	0001	0101
2	6	2	6	0010	0110
3	B	3	11	0011	1011
4	9	4	9	0100	1001
5	0	5	0	0101	0000
6	A	6	10	0110	1010
7	D	7	13	0111	1101
8	3	8	3	1000	0011
9	E	9	14	1001	1110
A	F	10	15	1010	1111
B	8	11	8	1011	1000
C	4	12	4	1100	0100
D	7	13	7	1101	0111
E	1	14	1	1110	0001
F	2	15	2	1111	0010

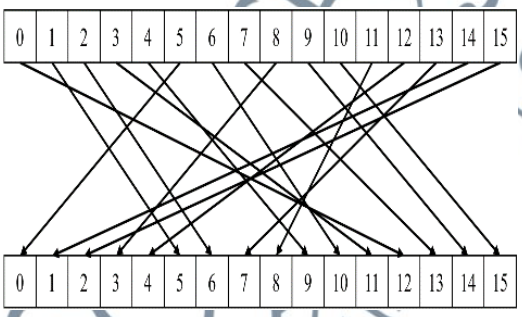


Figure 6.17: Output Distribution of S-box

The input for the S-box used in the *SubColumn* function is the output from the *AddRoundKey* defined as $Col_j = b_{3,j} \| b_{2,j} \| b_{1,j} \| b_{0,j}$ and the output is $S(Col_j) = c_{3,j} \| c_{2,j} \| c_{1,j} \| c_{0,j}$ for $15 \geq j \geq 0$ as depicted in Figure 6.18. Table 6.5 displays the input and output of the *SubColumn* function used in the encryption and decryption processes.

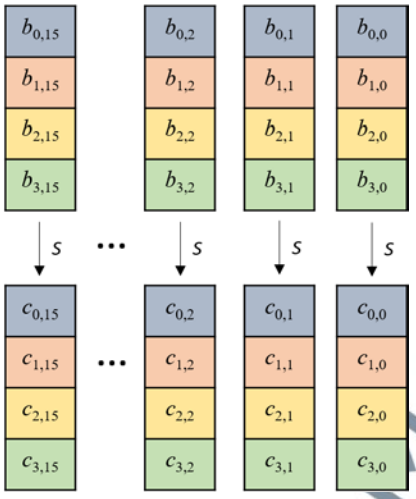


Figure 6.18: Sub Column

Table 6.5: Input and Output of Sub Column

	Input	Output
Encryption	Output of *AddRoundKey*	Input of *Double3DRotation*
Decryption	Output of *Double3DRotation*	Input of *AddRoundKey*

6.3.2.3 Double 3D Rotation

The formation of the *Double3DRotation* function is constructed using the *3DBitRotation_X-axis*, *AddRoundKey*, and *3DBitRotation_Z-axis*. This function is formulated based on the 3D rotation as specified in Section 6.2. In general, the input and output of the *Double3DRotation* function used in the encryption and decryption processes are displayed in Table 6.6.

Table 6.6: Input and Output of Double 3D Rotation

	Input	Output
Encryption	Output of *SubColumn*	Ciphertext
Decryption	Ciphertext	Input of *SubColumn*

6.3.2.3.1 3D Bit Rotation (*X*-axis)

The rotation formation of the *3DBitRotation_X-axis* function is rotated in a specific clockwise direction at the *X*-axis as follows:

i) $Slice_0$: Rotate 0^o (no rotation).

ii) $Slice_1$: Rotate 90^o.

iii) $Slice_2$: Rotate 180^o.

iv) $Slice_3$: Rotate 270^o.

Values of the *3DBitRotation_X-axis* function can be presented in the form of a permutation table as shown in Table 6.7. The output bits distribution of the *3DBitRotation_X-axis* can be observed in Figure 6.19.

Table 6.7: 3D Bit Rotation Permutation Table (*X*-axis)

Permutation Table (*X*-axis Rotation)							
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
28	24	20	16	29	25	21	17
30	26	22	18	31	27	23	19
47	46	45	44	43	42	41	40
39	38	37	36	35	34	33	32
51	55	59	63	50	54	58	62
49	53	57	61	48	52	56	60

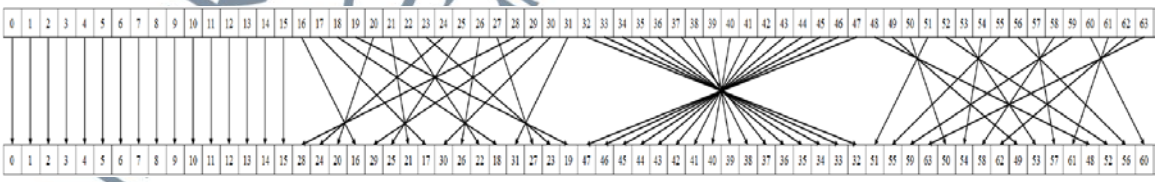


Figure 6.19: Distribution of Bit Rotation (*X*-axis) Output

The input for the *3DBitRotation_X-axis* is the output of the *SubColumn* function defined as $C = c_{3,15} \dots c_{3,0} \| c_{2,15} \dots c_{2,0} \| c_{1,15} \dots c_{1,0} \| c_{0,15} \dots c_{0,0}$ and the output is $P(C) = d_{3,15} \dots d_{3,0} \| d_{2,15} \dots d_{2,0} \| d_{1,15} \dots d_{1,0} \| d_{0,15} \dots d_{0,0}$ are depicted in Figure 6.20. Table 6.8 displays the input and output of the *3DBitRotation_X-axis* used in the encryption and decryption processes.

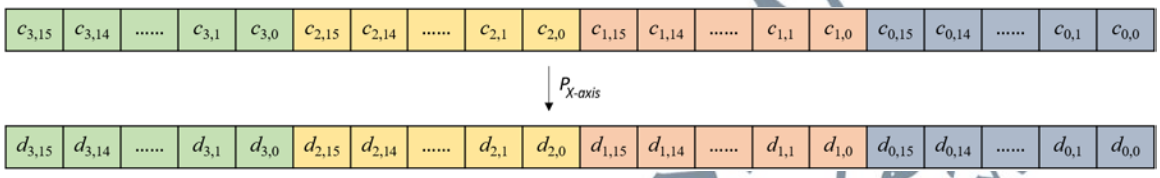


Figure 6.20: 3D Bit Rotation (*X*-axis)

Table 6.8: Input and Output of 3D Bit Rotation (*X*-axis)

	Input	**Output**
Encryption	Output of *SubColumn*	Input of *AddRoundKey*
Decryption	Output of *AddRoundKey*	Input of *SubColumn*

6.3.2.3.2 Add Round Key

In this operation, the output of the *3DBitRotation_X-axis* defined as $D = d_{3,15} \dots d_{3,0} \| d_{2,15} \dots d_{2,0} \| d_{1,15} \dots d_{1,0} \| d_{0,15} \dots d_{0,0}$ is XOR with the round key defined as $K = k_{3,15} \dots k_{3,0} \| k_{2,15} \dots k_{2,0} \| k_{1,15} \dots k_{1,0} \| k_{0,15} \dots k_{0,0}$ to produce the new cipher state as shown in Figure 6.21. For this function, the input and output of *AddRoundKey* used in the encryption and decryption processes are displayed in Table 6.9.

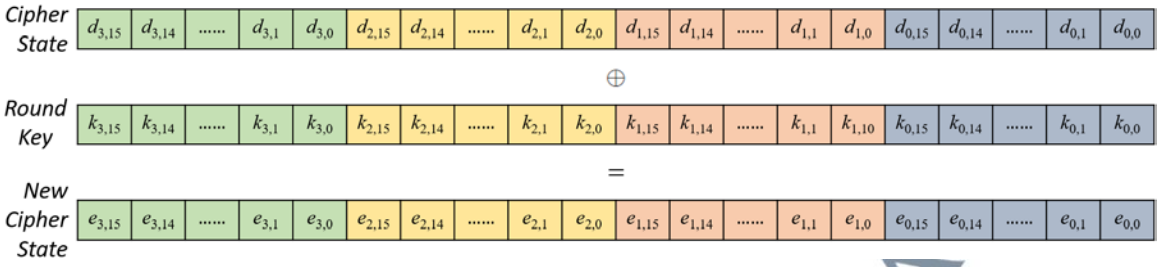


Figure 6.21: Add Round Key

Table 6.9: Input and Output of Add Round Key

	Input	Output
Encryption	Output of *3DBitRotation_X-axis*	Input of *3DBitRotation_Z-axis*
Decryption	Output of *3DBitRotation_Z-axis*	Input of *3DBitRotation_X-axis*

6.3.2.3.3 3D Bit Rotation (*Z*-axis)

The rotation formation of the *3DBitRotation_Z-axis* function is rotated in a specific clockwise direction at the *Z*-axis as follows:

i) $Slice_0$: Rotate 0^o (no rotation).

ii) $Slice_1$: Rotate 90^o.

iii) $Slice_2$: Rotate 180^o.

iv) $Slice_3$: Rotate 270^o.

The *3DBitRotation_Z-axis* values can be presented in the form of a permutation table as shown in Table 6.10. The output bits distribution of the *3DBitRotation_Z-axis* can be observed in Figure 6.22.

Table 6.10: 3D Bit Rotation Permutation Table (*Z*-axis)

Permutation Table (*Z*-axis Rotation)							
0	13	62	51	4	29	58	35
8	45	54	19	12	61	50	3
16	9	46	55	20	25	42	39
24	41	38	23	28	57	34	7
32	5	30	59	36	21	26	43
40	37	22	27	44	53	18	11
48	1	14	63	52	17	10	47
56	33	6	31	60	49	2	15

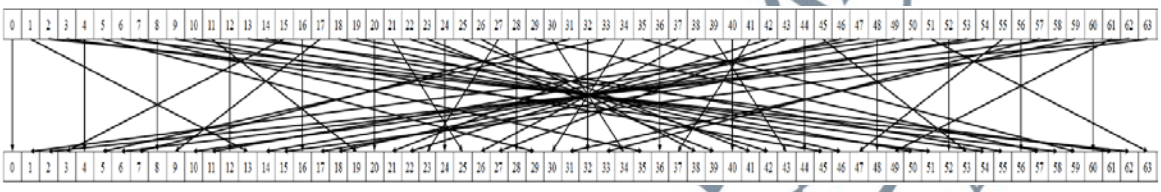


Figure 6.22: Distribution of Bit Rotation (Z-axis) Output

The input for the *3DBitRotation_Z-axis* is the output of the *AddRoundKey* defined as $E = e_{3,15} \dots e_{3,0} \| e_{2,15} \dots e_{2,0} \| e_{1,15} \dots e_{1,0} \| e_{0,15} \dots e_{0,0}$ and the output of this function produced the ciphertext defined as $P(E) = f_{3,15} \dots f_{3,0} \| f_{2,15} \dots f_{2,0} \| f_{1,15} \dots f_{1,0} \| f_{0,15} \dots f_{0,0}$ are depicted in Figure 6.23. For this function, the input and output of the *3DBitRotation_Z-axis* used in the encryption and decryption processes are displayed in Table 6.11.

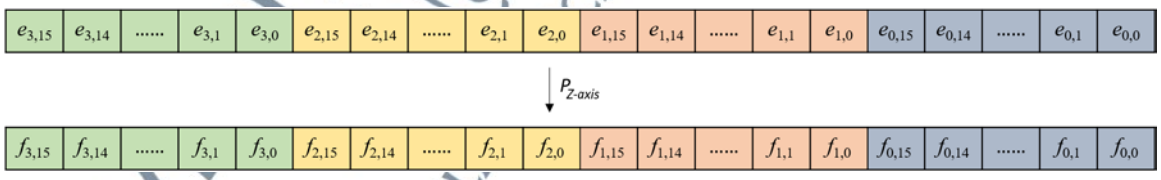


Figure 6.23: 3D Bit Rotation (*Z*-axis)

Table 6.11: Input and Output of 3D Bit Rotation (*Z*-axis)

	Input	Output
Encryption	Output of *AddRoundKey*	Ciphertext
Decryption	Ciphertext	Input of *AddRoundKey*

6.3.3 Key Schedule Algorithm

The round keys are derived from the secret key using the key schedule algorithm. In total, the number of round key bits is equal to the block length multiplied by the number of rounds plus one. (e.g., for a block length of 64 bits and 20 rounds, 1,344 round key bits are required). The *RoundKeyExtraction* process selected a 64 bits round key from the revised key after executing the *NonceXOR*, *KeySubColumn* and *RowTransformation* functions as shown in Figure 6.24.

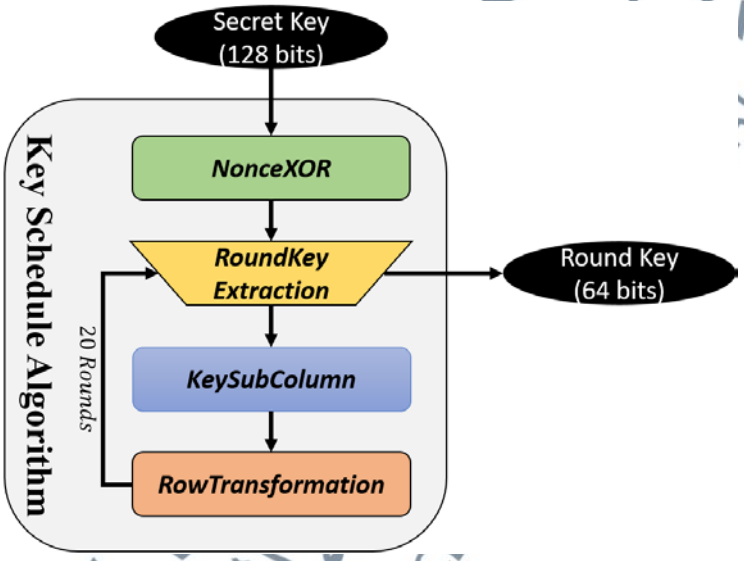


Figure 6.24: LAO-3D Key Schedule Process

6.3.3.1 NonceXOR

In the initial stage of the key schedule algorithm, the secret key defined as $S = s_{127}\|\ldots\|s_1\|s_0$ is XOR with a nonce defined as $N = n_{127}\|\ldots\|n_1\|n_0$ to obtain the key state as shown in Figure 6.25. The nonce is a random 128 bits data that is modifiable and embedded in the algorithm. In LAO-3D block cipher, the nonce is derived from the author's name as displayed in Table 6.12.

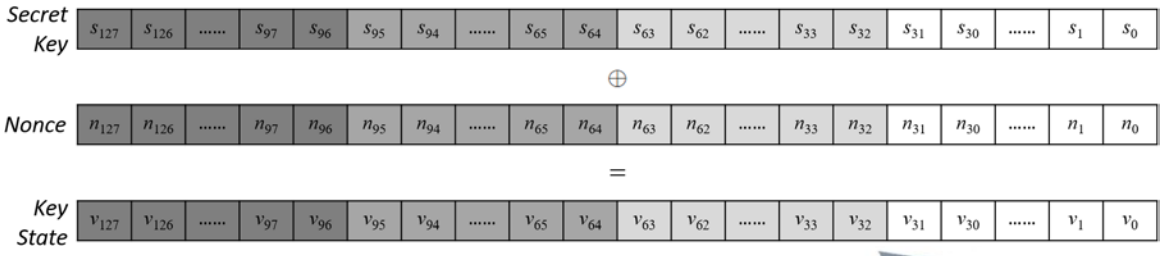


Figure 6.25: Key State

Table 6.12: Nonce

Nonce	
Character	ABDULALIFZAKARIA
Hexadecimal	414244554C414C49465A414B41524941
Binary	01000001010000100100010001010101 01001100010000010100110001001001 01000110010110100100000101001011 01000001010100100100100101000001

The purpose of adding the nonce in the key schedule algorithm is to increase the confusion property in the generation of the round keys. In addition, the nonce would remove the correlation between the secret key and the generated round keys. For implementation purposes, the nonce can be changed and agreed upon by the participating users such as a group of individuals or an organization. Using different nonce, a secret key should not be able to generate the same set of round keys, thus providing extra security to the users. For this function, the input and output of *NonceXOR* are displayed in Table 6.13.

Table 6.13: Input and Output of NonceXOR

Input	**Output**
Secret Key	Revised Key

6.3.3.2 Round Key Extraction

LAO-3D block cipher utilized a 128 bits key that generated a set of 21 round keys. The key schedule algorithm requires selecting 64 bits from the key state. Let $V = v_{127}\|\ldots\|v_1\|v_0$ defines the key state and each row is notated as the *RowKeys* as shown in Figure 6.26.

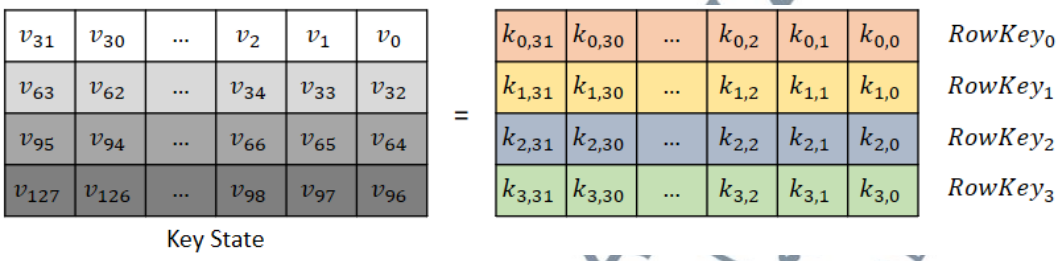


Figure 6.26: Row Key

16 rightmost columns of each *RowKey* are appended to produce the 64-bit of the i^{th} round key K_i at round i for $0 \leq i \leq 20$ as presented in Figure 6.27. The round keys are used in the *AddRoundKey* function during the encryption and decryption processes. For this function, the input and output of *RoundKeyExtraction* are displayed in Table 6.14.

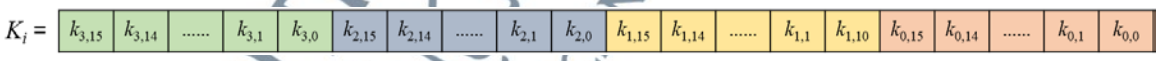


Figure 6.27: Round Key

Table 6.14: Input and Output of Round Key Extraction

Input	Output
Revised Key	Round Key

6.3.3.3 Key Sub Column

Upon completion of round key extraction, the round key value is revised in each round using the *KeySubColumn* function. Four uppermost rows and eight rightmost columns of the key state are reconstructed by using the same S-box from the *SubColumn* function in Section 6.3.2.2, i.e., $k'_{3,j}\|k'_{2,j}\|k'_{1,j}\|k'_{0,j} = S(k_{3,j}\|k_{2,j}\|k_{1,j}\|k_{0,j})$ for $7 \geq j \geq 0$ as depicted in Figure 6.28. For this function, the input and output of *KeySubColumn* are displayed in Table 6.15.

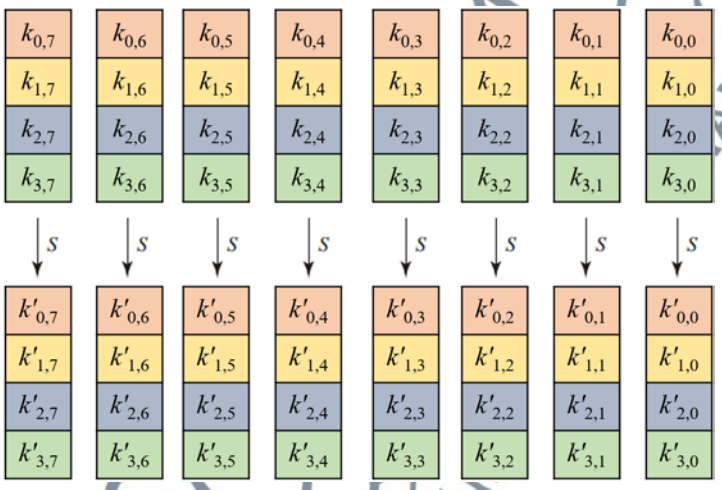


Figure 6.28: Key Sub Column

Table 6.15: Input and Output of Key Sub Column

Input	Output
Revised Key	Input of *RowTransformation*

6.3.3.4 Row Transformation

The new key state or known as the updated round key produced from the *KeySubColumn* is rearranged into *RowKeys* representation as shown in Figure 6.26. All *RowKeys* are revised using the *RowTransformation* function, i.e., $RowKey'_0 =$

$(RowKey_0 <<< 8) \oplus RowKey_1$, $RowKey'_1 = RowKey_2$, $RowKey'_2 = (RowKey_2 << < 16) \oplus RowKey_3$, and $RowKey'_3 = RowKey_0$ as displayed in Figure 6.29. For this function, the input and output of *RowTransformation* are displayed in Table 6.16.

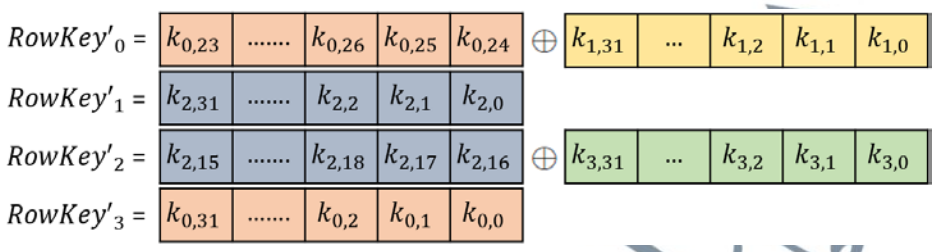


Figure 6.29: Row Transformation

Table 6.16: Input and Output of Row Transformation

Input	Output
Output of *KeySubColumn*	Input of Revised Key

6.4 Chapter Summary

This chapter described the formulation of the 3D rotation method that aims to provide confusion and diffusion properties in a block cipher algorithm. There are three innovations introduced in the new 3D cipher design. The first innovation of the 3D design is the implementation of a 3-dimensional cipher state in the form of cipher bits. Secondly, the 3D design does not require one to increase the block and key sizes of any block cipher that intend to adopt the 3D structure. Thirdly, the 3D rotation method applied double-axis rotations, which are the *X*-axis and *Z*-axis that are separated by an *AddRoundKey* function. This 3D rotation method is the main cryptographic component in the development of a new lightweight block cipher.

The base construction of LAO-3D is referred to the RECTANGLE block cipher. Introduction of the 3D rotation method probably will provide better confusion and diffusion properties to the original RECTANGLE algorithm. Double 3D rotations method in LAO-3D aimed to decrease the correlation between the plaintext and ciphertext, thus probably will contribute to the improvement of the randomness property.

Key schedule algorithm of LAO-3D block cipher probably will improve the previous version of RECTANGLE algorithm. Three modifications were made to the original algorithm that consist of reducing the number of round keys, increasing the length of constants, and improving the substitution component. The enhancements probably will solve the weak round key generation faced by the RECTANGLE key schedule algorithm.

For a better picture of the construction of LAO-3D lightweight block cipher, APPENDIX A, APPENDIX B, and APPENDIX C display each component of the key schedule, encryption, and decryption algorithms in detail. Test vectors of the algorithms are also provided in the appendices to help readers reconstruct the new block cipher.

The following Chapter 7 discusses the cryptanalysis of LAO-3D algorithm that includes Avalanche effect tests (correlation coefficient, bit error rate, and key sensitivity), randomness tests, and cryptanalysis attacks (linear and differential cryptanalysis) to observe the security of this lightweight block cipher.