Chapter 3  
A 3D Parity Bit Structure

3. Introduction

There is always a major role of data structure used in any program or software. “Bad programmers worry about the code, good programmers worry about data structures and their relationships” - Linus Trivolds. If a data structure used in a program is better so will be the program using it. This Chapter details about some structures that can be used for cryptographic algorithms, error detection/correction, and validation/verification process. This is not the end of the pursuit to find more, non-conventional structures like the ones introduced in further sections.

Structures are classified on the basis of the dimensions it uses. A direction that can be used to vary the specification of a structure is called a dimension. It was observed that researchers have proposed algorithms using one dimensional array, two dimensional arrays only. In this Chapter, three dimensional arrays are used for error detection/correction while transfer of information over the network.

3.1 Three dimensional Arrays

There are various structures that use more than two directions to vary its specifications. Structures with three such directions are called three dimensional structures. Arrays may have three dimensions in which an element uses three index values to access it. Generally, the co-ordinates are represented as X, Y, and Z co-ordinates in the co-ordinate space. Thus a three dimensional (3D) array may be viewed as an array of two dimensional arrays (matrices). Fig. 3.1 shows a

---

1 This Chapter is based on paper [88]: Pushpa R. Suri, Sukhvinder Singh Deora, “3D Parity Bit Structure: A novel technique to correct maximal number of bits in a simpler way”, International Journal of Computer Science and Information Security, Vol. 9 No. 8, August 2011, pp. 182-186
view of such array with some finite number of two-dimensional arrays (matrices) of fixed order.

![Diagram of a 3D array](image)

Fig. 3.1: A Three dimensional Array

### 3.1.1 Usage of Structures

The choice of structures in various programming problems cannot be ignored. The same is true for any other operations performed on the data during parity bit stuffing for error detection and correction, use of cryptography for data security. Further sections explain the use of 3D array in error detection and correction using parity bit.

### 3.2 Parity Bit

Information can be transmitted over the Internet using bits only. When a sender sends some information, it has to be converted into its binary representation before transferring it over the communication channel. At times, during transmission of the bits, the bit(s) can change, or can even get completely destroyed due to transmission errors that might occur due to signal distortion or attenuation also
known as noise. Techniques like checksums, weighted checksums, CRC codes etc are used for correcting such burst errors. A parity bit, an extra 0 or 1 bit that may be attached to the original bits may also be used to detect or correct errors [7, 112]. These extra bits may be used to correct the error bits in case of errors while transmission. Errors during transmission of bits are to be detected and/or corrected so that only the correct information is received by the receiver [13, 30, 89]. Claude D’Amours combine the two techniques rather than appending the parity bits at the end of the information sequence, the use of \((n-k)\) parity bits to select one of \(2^{n-k}\) orthogonal spreading sequences was suggested [28]. If wrong parity is found, the received data is discarded and retransmission is requested by the receiver [65, 66]. Parity bits may also be used to detect and correct such bit errors. Researchers Charles H. Bennett et.al. and G. Di Natale et.al. have described the use of parity bits in quantum cryptography and as S-box for AES algorithm respectively [23, 47]. This Chapter presents a simpler, effective and novel technique of using the parity bit coding using a 3D Array structure that may lead to correction of many bits in a simple way. The suggested method is beneficial for commercial usage too as it will reduce the number of retransmissions required.

3.3 The Proposed Parity Usage Technique

The encoding technique provides a new approach to the simplest and oldest error detection method using parity bit values. In this scheme, the binary data is taken in a 3D array by the sender of information. He adds binary digits called parity bits to indicate odd/even parity as decided prior to communication. The parity bit is appended to original data set at a fixed location in the 3D array bit structure. After receiver receives the information bits, they are tested for parity values to detect/correct errors, if any. The details are given in the next subsections.
3.3.1 Three Dimensional Array Block

Firstly, a three dimensional nxnxn sized array is taken to store the bits in row-wise storage as shown in Fig. 3.2 for a 3x3x3 array. The size nxnxn has been taken with the assumption that we have recognized that there were single or multi-bit errors in block size of n bits. If the information bits s are less than the smallest n such that $s \leq n^3$, the empty spaces are filled with bit stuffing. The bits added can be easily recognized as non-information bits.

![Fig. 3.2: Arrangement of a 3x3x3 Array Bit sequence](image)

3.3.2 Parity bits

Parity bits are added in each row and column as shown in Fig. 3.3. The Parity bits are shown as $P_{ijk}$ values for example fig. 3.2 i-th matrix’s j-th row and k-th column, for all matrix 1, matrix 2 and matrix 3 of 3x3x3 array of bits.

![Fig. 3.3: Parity Bits calculated Row and Column-wise](image)

The parity bit block can be then evaluated by finding $P_{ijk}$ of matrix i=4 (size of cubic array +1), combining the data bits of particular rows, columns and depth of all the metrics of 3D Array as detailed example in Fig. 3.4.
The formulas to be used to calculate the parity bits for a three dimensional Array $A$ of size $n \times n \times n$ is given by (3.1), (3.2) and (3.3). The formula for calculation of the parity bits for each row $j$ of $i$-th 2D subarray of $A$ is:

$$P_{i,j+1,k} = A_{i,j,1} \oplus A_{i,j,2} \oplus A_{i,j,3} \ldots \oplus A_{i,j,n} \quad \text{for } 1 \leq i \leq n, 1 \leq j \leq n \quad (3.1)$$

(3.2) shows the formula for calculation of the parity bits for each column $k$ of $i$-th two dimensional subarray of $A$.

$$P_{i+1,j,k} = A_{1,j,k} \oplus A_{2,j,k} \oplus A_{3,j,k} \ldots \oplus A_{n,j,k} \quad \text{for } 1 \leq j \leq n, 1 \leq k \leq n \quad (3.2)$$

Similarly, (3.3) is used to calculate parity bits in the $(i+1)$-th 2D subarray of $A$ for all positions $(j,k)$ in each 2D subarray from 1 to $n$.

$$P_{i+1,j,k} = A_{1,j,k} \oplus A_{2,j,k} \oplus A_{3,j,k} \ldots \oplus A_{n,j,k} \quad \text{for } 1 \leq j \leq n, 1 \leq k \leq n \quad (3.3)$$

### 3.3.3 Approach

The number of errors detected and/or corrected can be recorded for above scheme applying parity bits on a three dimensional array block. The approach is capable of correcting single bit error on all of the two dimensional matrices since it will mark a change in parity bit of all the three locations. Since, none of the parity will agree to error bit, the positions corresponding to erroneous parity bits is due to error at the location of intersection of those parity bit positions. Thus any single bit error
on the block is correctable since parity bits in all the three directions will become wrong and the position of the erroneous bit can be easily detected. The value of the parity bit at the identified location may be then toggled to get the corrected value. Consider the example, if bit 1 is erroneous (Fig. 3.5), then the parity bits $P_{114}$, $P_{141}$, $P_{411}$ will give us wrong parity. One can identify that the erroneous bit is the one that is at the intersection point of any two of the Parity line representations (red, green, blue) on some matrix corresponding to $P_{114}$, $P_{141}$ or $P_{411}$.

Even if it is assumed that at worst, maximum one parity bit of the structure corresponding to each bit can go wrong, such errors are correctible. As out of the three parity bit positions only two will be enough to identify the location of the wrong valued bit to be toggled. If there are one or more errors on one of the Parity line (red, green or blue) then also the errors will be detectable and correctable using two parity bits on other two parity line (Fig. 3.5).

Fig. 3.5 Example of Erroneous bit & corresponding Parity bits
However, if there are errors on the 3D array cube such that it forms a cube/cuboid shape, then those errors will become undetectable (Fig. 3.6). It is noteworthy that in such cases there will be even number of errors on all the parity lines that correspond to that particular bit position that goes undetected and are therefore is not correctable. In this case there is no hint of an error in the received bits as all the parity bits will appear to be correct in accordance to the data bits in the three dimensional array.

![Fig. 3.6: Cases that fall under CATEGORY III](image)

### 3.3.4 Error Categorization

All types of the bit errors may be categorized into three types:

- **Category I** – ‘DETECTABLE AND CORRECTABLE’
- **Category II** – ‘DETECTABLE BUT NOT-CORRECTABLE’
- **Category III** – ‘NOT-DETECTABLE’

Category I type errors are those in which it can be detected that error has occurred along with the position where it has occurred so that the bit can be toggled to
correct it. Category II type errors are those in which it can only be observed that something wrong has happened at some location of the bit set, but the location of error it is not traceable. In this case, the receiver will request the sender to re-send the information bits again since error has occurred but cannot be corrected at the receiver’s end. Category III type errors are the errors that can’t be detected as none of the parity bit hints about possible error occurrence.

3.3.5 Implementation

A program in C was developed for different size of three dimensional arrays that store some arbitrary bit values (information bits), a sample code is provided in Annexure A. The corresponding parity bits are then calculated according to (3.1), (3.2) and (3.3). It also takes some variable for counting the number of Category I errors that occurred in the simulation run and another variable that counts the number of errors of Category II until first error of Category III is encountered. The program increments value of variable that counts the errors of Category I or II as and when these occur in the simulation run. The random number generator function of C had been used to mark some arbitrary bit location as erroneous one by one. On error occurrence, the program checks whether that error falls under Category I / Category II / Category III. If error falls under Category I, its corresponding counter is incremented by one. If error falls under category II, its counter is incremented, as it can be detected that something wrong has occurred but not its location, still a re-send request may be used to have correct information again. The process stops on encountering first error of Category III as it will lead to erroneous information treated as correct at receiver’s end. In nutshell, the number of errors that can be detected and/or corrected are counted using the simulation runs of the concept. The C Program that counts the number of errors of such types detected/corrected is available in Annexure A. The average number of cases that can be detected/corrected using these simulation runs was evaluated against the number of parity bits used.
3.3.6 Test Results

The implementation in C program counted the errors which fall under the Category – I and after some modifications counted the errors which fall under the Category – II. The graph in Fig. 3.7 shows the average number of Category I and Category II type errors in case of particular sized 3D Array block. The graph shows increase in no. of case of both categories for a prime sized 3D Array block. This may be attributed towards the fact that if the errors are more random, the number of errors that fall under Category I and Category II be detectable in a prime sized 3D array block. Also it was noticed that there are some more good values near prime valued 3D array. However, for size of arrays that are divisible by small prime numbers like 2, 3 and 5, the number of errors of Category I or II decreases.

![Graph showing average number of detectable and correctable errors](image)

*Fig. 3.7: Number of Category I and Category II errors*

The average %-age of bits of Category I errors successfully corrected during the simulation runs is 2.49 % of the number of parity bits used and that of Category II
is 5.52% of number of parity bits used. The comparison between the number of parity bits used for a particular nxnxn sized three dimensional Array Block and the %-age of Category I and Category II errors is given in Fig. 3.8 below. Although the %-age of errors corrected decreases with increase in ‘n’ but overall behavior is such that most of the errors at various locations in the 3D Array can be detected and corrected.

![Comparison of %-age of errors of Category I and Category II errors in n-sized Cube](image)

*Fig. 3.8: %-age of Category I and Category II errors*

### 3.4 Conclusion

The results of the proposed scheme appear good as per expectations. The number of errors that can be detected increases with the increase in the size of three dimensional arrays. There has been increase in the average number of errors that can be detected / corrected for prime size three dimensional array blocks during simulation runs.
This experimental approach towards proposed 3D Array Parity Bit structure helped in drawing the following important inferences:

1. The 3D Array Block Parity bit scheme may be applied to any number of bits, as one can pad up the bits to complete the required number of bits for a complete 3D cubic array block. Hence, flexibility of size will enable use of this technique in different work areas of computer science.

2. The results for prime sized 3D Array Blocks were more convincing and can be used to detect and/or correct more than the average number of errors.

3. %-age of bit errors detectable and/or correctable reduces with increase in value of n for any nxnxn sized 3D Array. However, a good number of errors can be corrected/detected using this approach.

4. If all three parity bits are wrong, position of erroneous bit is definite.

5. If any two parity bits are wrong, it may be that one of the parity line contained even number of errors but the other two parity bits may be then used to identify the wrong bits and is again correctible.

6. If only one parity bit is wrong, it implies that there are even bit-errors on other two of the parity lines and so position of the error is not detectable, but it is inferred that there is some error. In such cases, the receiver may ask for re-send request to the sender, thereby avoiding making error.

3.5 Future Scope

The proposed approach may be applied to some other three dimensional structures or non-conventional structures to analyze the impact of this scheme error detecting and correcting property of parity bits using that structure. The concept looks simple yet good to work in situations when the approximate error bursts are known in advance so that one may decide about the structure to be used for appending and calculation of the parity bits.