Lightweight Error Detection Using Significant Bits

Michael Kennely[1], Scott Kerlin[2]
[1]Department of Mathematical Sciences, Gardner-Webb University
[2]Department of Computer Science, University of North Dakota

Overview

The increase in the demand for wireless data communications coupled with the growth of the small satellite field has led to the very real need to develop error detecting codes for a wide range of applications [1][2]. The proposed error detection algorithm features parity created rom the diagonals of the most significant bits to detect errors in a group of four bytes. This method distinguishes itself from other error detection schemes in that the redundancy overhead is only about 1/8th of the size of the data that it is being used on.

Unequal Diagonal Parity

The use of the diagonals to create parity stems from a chaotic encryption method, which was adopted to offer a resilience to burst errors through more random sampling during parity bit creation[3]. Using only the most significant bits to create the parity, particularly for images, allows the user to find errors where they affect the data the most. The algorithm is as follows:

. Read in four bytes
. Stack the bytes horizontally
. Create a 4x4 matrix using the most significant four bits in each byte
. Traverse each diagonal and xor each position of the diagonal to find the parity
. Insert the four parity bits after each 4 bytes of data

The typical burst error pattern that is found in space is that two consecutive bits will be flipped[4]. Using diagonal parity allows the user to combat this issue as consecutive bits have no influence on the same parity bit. Ideally, this should enable the user to obtain a lower bit error rate by reducing the signal strength throughput to a lesser degree than other techniques, while still providing some data integrity[5].

Graphs 1 & 2

Figures 1 and 2 are graphs that display how much time was taken to compute the parity and then check the parity for various JPEG files. Lena is 41.5 kb with dimensions of 512x512. Northern Lights is 390 kb with dimensions of 5016x3084. Cloud Strife is 670 kb with of dimensions 2340x1326. These graphs are used to show how time efficiency fluctuated as file sizes were changed.

Background

In the field of error detection and correction (EDAC) three algorithms are dominant: Hamming Codes, Reed-Solomon Codes, and Turbo Codes. A brief description of each will be given. Hamming codes are widely used in EDAC. They are popular because the encoding and decoding processes are easily implemented; however, they require a channel or environment that has a minimum amount of noise to be effective. Burst errors also prove to be problematic for Hamming Codes[3]. Reed Solomon (RS) codes can be found in communications and data storage. RS codes provide defense against burst errors, but entail a more strenuous implementation. In deep space communications, Turbo Codes are popular EDAC methods. The drawback to Turbo Codes is that they are costly to implement[1]. The goal of this research was to find a lightweight, yet effective method that took the best of each method mentioned above while minimizing the drawbacks of each.

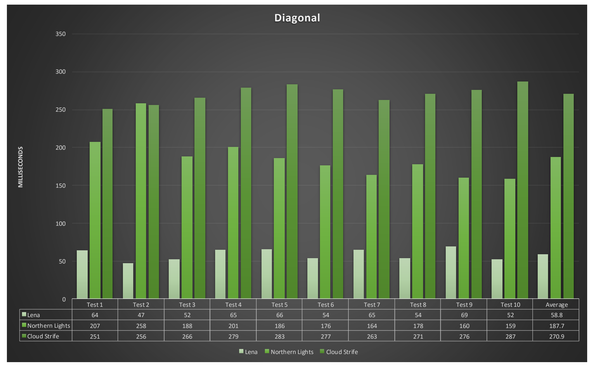

Figure 1: Graph of encoding/decoding time of Diagonal Parity

Figure 2: Graph of encoding/decoding time of Hamming Codes

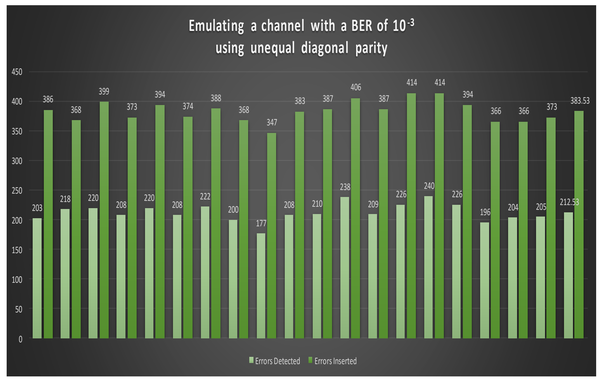

Figure 3: Graph of errors detected/inserted with a BER of 10^{-3}

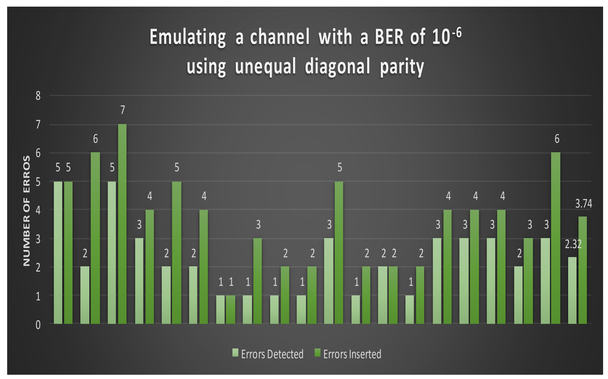

Figure 4: Graph of errors detected/inserted with a BER of 10^{-6}

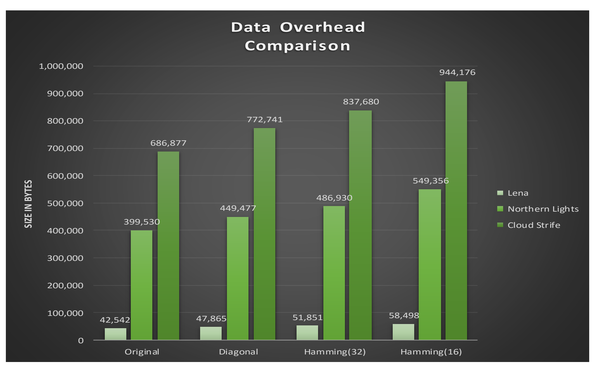

Figure 5: Graph displaying data overhead

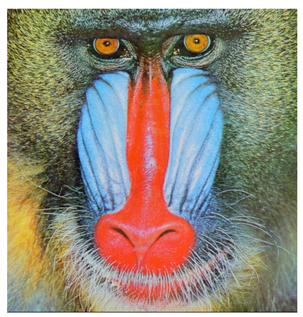

Figure 6: Image of a baboon
http://www.sci.utah.edu/~cscheid/spr05/imageprocessing/project4/imgs/

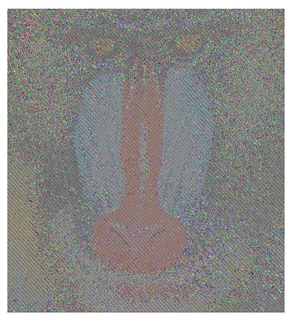

Figure 7: Image with least significant 4 bits in each byte flipped

Conclusions

The data gathered from the various tests that the proposed algorithm was put through suggests that this algorithm is superior to Hamming Codes when it comes to how large the data overhead is. Whether using a code word size of 16 bits or of 32 bits, this algorithm used far less space than Hamming Codes did, especially as file sizes grew larger. However, Hamming codes are assured to detect any single error in a code word[3]. At this time, this algorithm isn't suitable for any sort of error detection, but given a greater focus towards specific image formats it can prove to be potent in environments where memory overhead is of a high concern and data quality can suffer to a degree. In conclusion, Hamming Codes appear to be far more useful than the proposed algorithm in general, but in a resource starved environment, they could be edged out in favor of an enhancement on this algorithm.

Future Work

To further the research presented on this poster, an error correcting component will need to be added to this algorithm. Adjustments for various image formats will also be required as different encoding schemes leaves the idea of the most significant bits ambiguous. Providing support particularly for JPEG and PNG images is a goal.

References

1. Mitchell, G. 2009. Investigation of Hamming, Reed-Solomon, and Turbo Error Correcting Codes. U.S. Army Research Laboratory.
2. Ødegaard, K. 2013. Error Detection and Correction for Low-Cost Nano Satellites. Norwegian University of Science and Technology.
3. Huang, X., Ye, G., Wong, K. 2013. Chaotic Image Encryption Algorithm Based on Circulant Operation. Abstract and Applied Analysis, Vol. 2013, Article ID 384067
4. Hodgart, S., Tiggeler, H. 2000. A(16,8) Error Correcting Code (T=2) for Critical Memory Applications. European Space Agency.
5. Wertz, J., Everett , D., Puschell, J. 2011. Space Mission Engineering: The New SMAD. Pg. 473-480. Space Technology Library.

Acknowledgements

This research was funded by the U.S. National Science Foundation (NSF Award #1359224) with support from the U.S. Department of Defense. I would like to thank Scott Kerlin for advice and direction over the past eight weeks. Thank you Zach Chavis for your consultations.