Journal of Business and Management Sciences, 2018, Vol. 6, No. 3, 70-75
Available online at http://pubs.sciepub.com/jbms/6/3/1

DOI:10.12691/jbms-6-3-1

Reed-Solomon Based Bar Code Character Substitution Rates

Kevin Berisso*

Department of Engineering Technology, University of Memphis, Memphis USA
*Corresponding author: kberisso@memphis.edu

Abstract The use of Reed-Solomon error correction (RSEC) in bar code symbologies has been around since the early 1990s, but there hasn't been an in depth, publicly accessible study done on their resistance to character substitution errors since the original studies done in the early 1990s at Ohio University. This article reports on the test results from the scanning of more 23 million scans resulting in more than 2.39 billion characters across four different RSEC enabled bar code symbologies (Data Matrix, QR code, PDF417 and Aztec Code) with five different scanners. The results show that the RSEC enabled symbologies are capable of achieving at least a 1 in 797 million error rate, allowing for their use in instances where decoded data accuracy is imperative.

Keywords: *2D symbology, bar codes, data matrix, QR code, reed-solomon*

Cite This Article: Kevin Berisso, "Reed-Solomon Based Bar Code Character Substitution Rates." *Journal of Business and Management Sciences*, vol. 6, no. 3 (2018): 70-75. doi: 10.12691/jbms-6-3-1.

1. Introduction

Within the automatic identification industry, the use of bar code symbologies (the various types of bar codes or 'languages') that include Reed-Solomon error correction (RSEC) is often taken as a given for applications that require a higher degree of data confidence. The reasoning is that the Reed-Solomon error correction methodology is known to be a proven, robust solution that is used across multiple industries. Data substitution errors are instances where the bar code scanner processes a bar code symbol and returns an incorrect answer that is of the correct length and formatting, but with one or more incorrect characters or digits (e.g. "1235" instead of "1234"). However, if the reader were to specifically look for published evidence of the applicability of Reed-Solomon based 2D symbologies for industries with a zero tolerance of character substitutions exist, there exists little or no published evidence. In fact, the only known publicly available discussion of 2D symbology data substitution rates is an unpublished Ohio University report by Fales & Vincent [1] from 1993 in which they scanned approximately 62 million characters between the Data Matrix and PDF417 symbologies. Considered a seminal work for its time, the current level of 2D symbology usage today implies that this work is in dire need of an update.

In 2013 the Federal Drug Administration (FDA) released a final rule that required the unique identification of medical devices through their distribution and use. Known as the Unique Device Identification (UDI) rule, this FDA mandate requires all devices not under exemption to carry a label that contains both a human readable component and in an automatic identification and data capture (AIDC) form [2]. In the rule, the specification of specific technologies was avoided. However, due to the required data content and the nominal size of many of the medical devices that would have to comply, smaller high density bar code symbologies would have to be incorporated. And because of the sensitivity to data accuracy within the medical industry, the implicit use of a symbology with error correction was functionally build into the rule.

The FDA's UDI rule is not the first government mandate that implicitly or explicitly require 2D symbologies. As early as 2003, the U.S. Department of Defense required Item Unique Identification (IUID) marking on all serially managed items via MIL-STD-130 (which started out as Department of Defense Policy 4140.01) where all items above a specified value or that were otherwise specified received a unique identifier. Within the standard, Data Matrix is specifically listed as an acceptable method of marking required items.

The use of 2D symbols is also an accepted practice within the consumer goods supply chain. GS1, a global business standards organization that enables companies "…to identify, capture and share information smoothly, creating a common language that underpins systems and processes all over the world." [3], adopted the Data Matrix symbology in 2006 as a way to allow members to mark items for the healthcare industry. In fact, one only has to look to their cans of soup, shampoo or medication to see the adoption of Data Matrix as a tracking or validation tool.

And finally there is the airline industry. The use of PDF417 and Aztec Code have been an integral part of the airline industry since the early 2000's as a way to encode passenger data on both paper tickets and cell phone-based e-tickets. Since the passenger's name and flight information is encoded, a single incorrect character has the potential to disrupt or even prevent a traveler's passage through security and onto the plane.

What all of these groups have in common is the need to have a symbology that is resistant to physical damage (erasures, additional marks inside the symbol, etc.) ensuring that the data is still available and still accurate. 2D symbologies such as Aztec Code, Data Matrix, Han Xin, PDF417, and QR Code provide this level of protection and data integrity insurance. However, as previously mentioned, there currently exists a lack of empirical evidence to this effect. While a seminal piece of work for its time, the 1993 study by Ohio University has two short-comings. This first is that the study is over 24 years old; there have been numerous significant improvements in micro-computer processing and imaging since the tests were conducted. The second issue is that there has been a significant increase in the usage of the tested symbologies. The 1993, results indicated that in at a worst case there could be a one in 10.5 million probability of a character substitution error and a best case of one in 613 million. However, because of the data set size, 31 million characters scanned over 622,080 instances, this result was only a statistical probability based on the confidence interval of the test.

The goal of this research was to determine if the one in 10.5 million character substitution probability was realistic and to hopefully provide the AIDC industry with a new benchmark on the robustness of Reed-Solomon based symbols. This research is not looking at the comparative performance of the scanners used nor the symbologies but instead is concentrating on the ability of correctly printed, non-damaged symbols to consistently and reliably result in accurate data. The reasoning for this limitation is that to compare scanners or symbologies would actually be testing the various image processing and decode methodologies used by the various scanner manufacturers, not the error correction robustness; something that is outside the scope of the papers intent.

2. 2D Bar Code Symbologies

To the lay person, all bar codes are often grouped together. However, to the AIDC expert, there are often profound differences that allow the various symbologies to be used to solve specific problems. For example, the use of QR Codes in an on-demand high speed printing process would make as much sense as using a hammer to drive screws would be to the carpenter. Additionally, not all symbologies include error correction, making their use a poor choice when encoding data in environments that are prone to label damage or are in need of a high level of data accuracy confidence. For example, in his 1991 report, Dr. Fales reported a worst case character substitution rate for the UPC-A symbology of 1 in 394,003 characters at a 95% confidence interval. While Code 128 did better, 1 in 2,764,151 characters at a 95% confidence interval, the test showed that character substitutions could and did occur [4].

2.1. Terminology

Within the AIDC industry, there are some common terms that are used that the lay person may not know or may incorrectly use.

1) 2D symbology – A symbology that encodes data in two dimensions. Both stacked and matrix symbologies are often categorized together as 2D symbologies.

2) Codeword – "symbol character value, an intermediate level of coding between source data and the graphical encodation in the symbol" [5]

3) Error correction codeword (ECC) – Codewords that are specifically used for error correction instead of data encoding.

4) Linear symbology – A symbology that encodes data in one direction only.

5) Matrix symbology – A symbology that encodes data in a matrix of modules that are either on or off.

6) Module – A single cell or element within a grid that represents a single data bit for a matrix symbology.

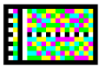
7) Stacked symbology – A symbology that encodes data in multiple rows that often look like linear symbologies stacked upon each other.

8) Substitution error – An instance where one or more decoded characters differ from the encoded character(s).

9) Symbol – A specific instance of a bar code that has been encoded with data following a bar code symbology. An analogy would be a word written in a specific language.

10) Symbology – A standard way of encoding data in machine readable form. An analogy would be a language (e.g. French, Spanish, Korean).

Table 1. Some common 2D symbologies in use

Symbology	Example	Comments
Aztec Code		Currently used in the airline industry.
Data Matrix		Most commonly seen 2D symbology when manufacturing is included.
Dot Code		While an open standard, only the tobacco industry has seen any level of adoption.
Han Xin		China is the primary user at the moment.
Maxicode		Maxicode is only used by UPS.
PDF417		PDF417 is a stacked symbology, where the rows in the symbol contain varying information.
QR Code		Commonly seen in mobile applications, arguably the most recognizable 2D symbology.
Ultracode		While an open standard, Ultracode has not seen large scale adoption to date.

11) X Dimension – The module size for matrix symbologies and the smallest bar or space width for linear and stacked symbologies. An analogy would be the font size used for the symbol.

2.2. Types of Symbologies

In general there are three primary types of bar codes in use; linear, stacked and matrix. In traditional linear

symbologies (e.g. EAN/UPC), the height is referred to as vertical redundancy. No additional information is stored in the vertical height of the symbol. Instead, the vertical height exists to make the scanning process easier. Stacked symbologies use stacked rows of linear type bars to encode data and matrix symbologies use the presence or absence of modules within a grid to encode the binary data. For the purposes of this paper, stacked and matrix symbologies have been grouped together under the heading 2D symbologies.

While there are numerous stacked and matrix based bar code symbologies in existence (see Table 1), this paper is concentrating on what are arguably four of the five most commonly seen symbologies; Aztec Code, Data Matrix, PDF417 and QR Code. Han Xin is a newer symbology out of China that is seeing an expanding role in some areas of the supply chain. However, due to its newness, not all scanners support it, limiting its use.

What the four selected symbologies have in common is wide-spread adoption and the inclusion of error correction. These symbologies take the data that is to be encoded and convert the information into a series of codewords. These codewords are then mapped to the required patterns for printing.

The bar code symbol also will include some method of indicating an origin or start and stop location (e.g. QR Codes use the three distinct finder patterns, in the shape of squares with bounding boxes as seen in Figure 1). This finder pattern helps the decoder to determine the extents of the symbol and makes it easier for the imaging processor to know what is and is not part of the bar code.

Figure 1. Finder patterns (black areas) for Aztec Code (left), Data Matrix (middle-left) and QR Code (right). PDF417 (middle-right) uses start (left side) and stop (right side) patterns

Of the four symbologies discussed, only Data Matrix has a fixed level of error correction. The others have some level of user selectability when it comes to the amount of error correction that is applied to a given symbol. Additionally, Data Matrix is the most efficient of the symbologies. Given the same data and the same X Dimension, Aztec Code requires 1.7 times more space, QR Code requires 3.2 times more space and PDF417 requires 29.2 times more space when encoded at their minimal error correction levels.

2.3. Reed-Solomon Error Correction

RSEC is a methodology for auto-correcting erroneously decoded messages. Originally developed for the telecommunications industry, RSEC that has been widely adopted as an error correction solution across multiple industries. With a solid history of use within the communications disciplines, RSEC was adopted by the AIDC industry when 2D symbologies were developed as a method for addressing the potential of having missing modules (erasures) or instances where codewords were incorrectly decoded (errors) [6,7,8].

While a full discussion of the mechanics of the RSEC methodology is beyond the scope of this article, the general process is to pass the data to be encoded though a series of steps that convert the source data into a series of integer based code words that is constrained to a mathematical field of a predetermined prime size (the Galois Field). These codewords are then combined with the appropriate Galois Field based RSEC polynomial for the desired number of error correction code words. The resulting error correction code words are embedded within the message and everything is sent to the receiving device; which in the case of bar codes is the logic that converts the code words into a series of on or off modules within the symbol [9].

When a bar code symbol is read, the decoder will reverse the process since the RSEC polynomial is known and the number of codewords will be either explicitly or implicitly defined. If a remainder exists after dividing the code word(s) into the generating polynomial then errors have been detected and the corrected code words can be calculated, allowing for the recovery of the original message.

The four symbologies selected for this study use a variety of Galois Field values and differing prime modulus numbers. The exact GF values and prime modulus numbers are listed in the symbology's respective AIM or ISO/IEC standards and are based on the idiosyncrasies of the individual symbology.

3. Methodology

The physical testing setup for this paper was relatively simple. Bar code scanners were attached to a simple gantry frame that allowed each scanner to be individually positioned as shown in Figure 2. The individual symbols used were printed on a standard laser printer from the Bartender Designer 2016 R3 software from Seagull Scientific and the symbols were attached to simple medal plates with magnets. The plates were attached to a reciprocating table that was driven by a variable speed motor. The table was set to reciprocate at a rate of 2 Hz. The table was located such that at the end of each stroke, the symbols were moved within the scan field of the bar code scanners.

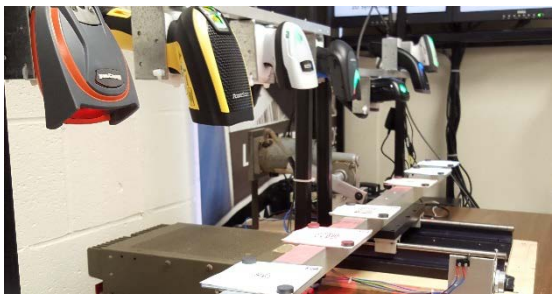


Figure 2. Physical testing setup

The symbols that were used contained various random data that was in five different formats (boarding passes, GS1 data formats, FDA UDI encoded using GS1 data constructs, random numbers, alphanumeric data). All symbols were printed in their correct format without any

manufactured defects (e.g. erasures, additions, skewing). It was decided a priori that for this specific test a stressing of the individual decode algorithms that had been implemented by the various scanner manufacturers would not be conducted. Instead, this test was intended to determine if random character substitutions would occur as was found with most of the traditional linear symbologies (e.g. Code 39, EAN/UPC) in the studies done by the Center for Automatic Identification at Ohio University in the early 1990's [2].

The bar code scanners were connected via RS-232 serial interfaces that fed directly into the computer (instead of using USB to RS-232 adapters) so as to remove any potential transmission errors due to the USB interface or the existence of multiple USB to Serial adapters on the same computer. The specific bar code scanners used were all hand-held scanners that were placed in "presentation mode" which allows the scanner to continuously decode symbols that are within the scanner's field of view without the need for a physical trigger pull. The selection of scanners was random in that specific scanners with a priori capabilities were not identified. Instead, scanners that were readily available in the lab, or that could be obtained via donations were used. All scanners had only the four symbologies being tested enabled and were configured to transmit the AIM Symbology Identifier (a "]" plus two additional characters) as a prefix and a carriage return and or line feed as a terminator.

Custom software written in C# using Microsoft's Visual Studio was developed to capture the serial port data and insert it into a SQL Server database that was maintained on a separate computer. The software captured the scanned data and recorded the scanner station. It should be noted that the data reported in this report is actually from a second data collection run. In the first data collection run, an unacceptable number of transmission error occurred due to the method in which the custom software processed data. It was discovered that when searching through the serial buffer data for the termination characters, casting the hexadecimal ASCII values to characters instead of using the escape sequence (e.g. "(char)0x000A" instead of "\r") significantly improved processing time and reduced the number of transmission based errors within the software.

4. Discussion

Over the course of the research, a total of 23,869,258 scans were collected for a total of 2,391,925,961 characters. 55 unique symbols were processed. The results of the scans were grouped by encoded data and scanner and three types of data were searched for; correct data (99.9967%), incorrect data due to transmission errors (0.0033%) and incorrect data due to character substation errors (0.0000%).

4.1. Correct Data

Correct data was identified within the database as data strings that matched the encoded data in the various symbols. In some instances the custom software left a null character at the end of an otherwise valid string. For these instances, the null character was stripped away and the resulting data string was added to the count for those instances where the null character was already stripped.

Table 2. Breakdown Of Symbologies Scanned

Symbology	Total Scans	Characters Scanned
Aztec Code	4,216,918	606,895,007
Data Matrix	7,084,222	603,184,399
PDF417	6,265,983	551,891,004
QR Code	6,302,135	629,955,551

Table 2 shows the breakdown of the symbologies scanned and the relevant overall number of symbol scans and characters scanned. The quantities listed are of the scans that were indicated as good. Scans that were ultimately categorized as having transmission errors were left out of the totals since the character counts would have skewed the results.

4.2. Transmission Errors

Despite the efforts taken to eliminate transmission errors between the scanners and the software, some still occurred. The errors that did occur can be grouped into three types; dropped characters, truncated data, and corrupted data. For the purpose of this paper, dropped characters are instances where no more than three characters were missing from the data stream. Truncated data is being defined as more than three characters were missing from the data stream but where there was no other issues with the data stream. Corrupted data is defined as instances where non-printable ASCII character appeared in the data stream or where additional unexpected data

By far, the most frequent occurrence of transmission error was the dropping of one or more characters, 514 occurrences. The majority of the time the dropped character was the carriage return and/or line feed that indicated the end of transmission of a data set. The reason for this is unknown, but at a rate of 21.5 per million transmissions it is assumed that the problem was either in the custom software or due to minor glitches in the serial buffers on the computer. However, this type of error would occur infrequently enough that it shouldn't be of major concern to users as properly written programs will provide a level of data validation prior to the consumption of the data.

There were 60 instances where the data transmission appeared to be significantly truncated. Once again, the cause is unknown but it was assumed that the causes were either in the custom software or due to minor glitches in the serial buffers on the computer.

The third type of transmission error exhibited itself as corrupted data. Two types of corrupted data were observed during the testing; truncated and expanded. In terms of unique data streams, there were a significantly higher number of truncated transmission errors. However, most of these occurred only once or twice.

The corrupted data that resulted in truncated data primarily manifested itself as non-printable ASCII characters. While the exact cause is unknown, it is believed that there was some sort of interruption in the bit stream that caused some sort of bit shift. There were,

however, 116 instances where printable ASCII characters were introduced into the data stream (e.g. "?", "@").

Conversely, there were relatively few unique instances of expanded data streams, but when they did occur, there were thousands of instances. In terms of the number of times it occurred, the most prominent problem (over 66 thousand occurrences) was the addition of one or more instances of the string "]C0?" with an additional number after the question mark (0, 6 or 8). It is unknown why this occurred. The string "]C0" indicates that the symbology scanned is Code 128. However, this is occurred on a single scanner despite Code 128 being turned off in the scanner's configuration. There were also six instances where a question mark was inserted into the data stream as the first character instead of the "]" that was supposed to be there. However, despite the additional character(s) added to the data stream in all of these cases, the encoded data was not impacted. The manufacturer of the scanner was contacted and it was determined that there was a bug in the scanner's firmware that resulted in the unintentional additional characters. This was proved when an updated firmware version was loaded into the scanner and the problems went away.

The other unexplained occurrence was where a single scanner (which was different from the one previously discussed) inexplicitly substituted a random sequence of numbers into the middle of a string of 87 zeros a total of 83 times. The result was that instead of transmitting

"]Q1100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000006"

the scanner would transmit

"]Q1100000000000000000000063894522544000000000000000000000000000000000000000000000000000000000000000000000000"

to the data collection software. The symbol in question was a QR Code with the error correction level set to Q or H. When the same scanner processed the same data but encoded at an error correction level of M or L, the problem did not manifest. After discussions with the manufacturer, it was determined that the cause was a bug in the firmware, not in the symbology because a) no other scanner exhibited the behavior – including a different model scanner from the same manufacturer, b) that the final character in the incorrect string was not correct and c) that despite numerous attempts, the researcher was not able to mathematically show how the change in data would have been possible within the confines of the Reed-Solomon error correction process. Furthermore, it should be noted that when other over-corrected symbologies were tested on the scanner in question, no instances of invalid data were observed, adding to the belief that the problem was isolated to the specific version of firmware.

4.3. Substitution Errors

Of the 2,391,925,961 characters scanned over 55 symbols, there were zero identified instances where a character substitution occurred after 23,869,258 scans. There were 83 instances where one scanner produced data that appeared similar to a character substitution error. However, as previously discussed, these occurrences were ultimately classified as firmware specific error and were not attributable to the symbologies or error correction.

4.4. Statistical Conclusion

Based on van Belle, "given no observed events in *n* trials, a 95% upper bound on the rate of occurrence is 3/n" [10]. Thus, for the generated data, it can be said with a 95% confidence that RSEC will provide protection to at least a 1 in 797 million character substitution rate based on the data collected.

5. Conclusions

The results from this study show that at a 95% confidence level, bar code symbologies that employ RSEC have at character substitution resistance that results in an occurrence rate that is at least 1 in 797 million. Instances of character substitution found outside the lab, in symbols that are in good condition without physical damage, will most likely be due to either transmission errors or bugs in the firmware of the bar code scanner producing the data. And even then, the likelihood that a user will see data similar to what was used to provoke the errors found during this test are highly unlikely – how often would a real-world business case require the encoding of 87 consecutive zeros.

During the course of the research it became apparent that solution providers will need to ensure that they are accessing data in a manner that best suits the individual application. A careful implementation of any serial port based data collection will reduce the number of transmission errors and the proper validation of incoming data will further ensure that invalid data doesn't make its way into the system.

This study shows that RSEC based symbologies are sufficiently immune to character substitutions errors such that the adoption of these symbologies can be comfortably accomplished during the compliance process to such initiatives as the FDA UDI or the Department of Defense IUID programs. However, there is still room for additional improvements to the body of knowledge about RSEC based symbologies. Testing of the error correction capabilities due to erasures or symbol errors should be done to help determine the true level of robustness of these symbologies. Additionally, testing of the various bar code scanner interfaces should be done to help determine if an avoidance of specific interfaces should occur if the reduction in the number of transmission errors is desired.

Acknowledgements

This paper would not have been possible without the support and advice of the GS1 Barcode and Identification Technical Group and the AIM Technical Symbology Committee. Additional thanks are owed to Cognex, Datalogic, Honeywell Scanning and Mobile Productivity, and Zebra for their outstanding technical support in configuring and optimizing the settings for their respective devices. For further information about this study please visit www.memphis.edu/autoid/research.

References

[1] J. F. Fales and R. S. Vincent, “Datamatrix and PDF417 Data Integrity Test,” unpublished report to Oak Ridge National Laboratory dated Oct. 1993.

[2] FDA Final Rule – Federal Register Vol 78, No 185, Sept 24, 2013 Available: https://www.gpo.gov/fdsys/pkg/FR-2013-09-24/pdf/2013-23059.pdf.

[3] GS1.org, ‘Home Page’, 2017. [Online]. Available: https://www.gs1.org/. [Accessed 19- May- 2017].

[4] J. F. Fales, “Code 16K and Code 49 Data Integrity Test,” unpublished report to AIM USA dated Dec, 1991.

[5] Information technology — Automatic identification and data capture techniques — Data Matrix bar code symbology specification, ISO/IEC 16022, 2006-09-15.

[6] Li, L., Qiu, J., Lu, J. and Chang, C. “An aesthetic QR code solution based on error correction mechanism”, *Journal of Systems and Software*, 116, ppg. 85-94, 2016.

[7] Wicker, S. B. and Bhargava, V. K. Reed-Solomon Codes and Their Applications. Wiley-IEEE Press. 1999.

[8] Kato, H. and Tan, K. T. “Pervasive 2D Barcodes for Camera Phone Applications”, *IEEE Pervasive Computing*, 6 (4), pp. 76-85, Oct 2007.

[9] Plank, J. S. “A Tutorial on Reed–Solomon Coding for Fault-Tolerance in RAID-like Systems”, *Software—Practice And Experience*, 27(9), ppgs. 995–1012, SEPTEMBER 1997.

[10] G. van Belle, "*The Rule of Threes for 95% Upper Bounds.*" in *Statistical Rules of Thumb*, 2nd ed. Hoboken, N.J., U.S.A: Wiley, 2008, ch. 2, pp. *49-50*.