

# Provably Good Codes for Hash Function Design

Charanjit S. Jutla  
IBM Thomas J. Watson Research Center  
csjutla@watson.ibm.com

Anindya C. Patthak\*  
University of Texas at Austin  
anindya@cs.utexas.edu

## Abstract

We develop a new technique to lower bound the minimum distance of quasi-cyclic codes with large dimension by reducing the problem to lower bounding the minimum distance of a few significantly smaller dimensional codes. Using this technique, we prove that a code which is similar to the SHA-1 message expansion code has minimum distance 82, and that too in just the last 64 of the 80 expanded words. Further the minimum weight in the last 60 words (last 48 words) is at least 75 (52 respectively). We expect our technique to be helpful in designing future practical collision-resistant hash functions. We also use the technique to find the minimum weight of the SHA-1 code (25 in last 60 words), which was an open problem.

**keywords:** linear codes, minimum distance, collision-resistant hash functions, SHA-1

## 1 Introduction

Recall the SHA-1 message expansion code which is a binary linear code of dimension 512: the 512 information bits are packed into 16 32-bit words  $\langle W_0, \dots, W_{15} \rangle$ , and 64 additional words are generated by the recurrence:

$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1 \quad \text{for } i = 16, \dots, 79 \quad (1)$$

The 80 words  $\langle W_0, \dots, W_{79} \rangle$  can be seen as constituting a code-word in a linear code over  $\mathbb{F}_2$  with the above parity check equations. Unfortunately, this code has a minimum distance or weight of no more than 44. Further, the weight restricted to the last 60 words is only 25. This has been exploited in [WYY05c] to give a differential attack on SHA-1 with complexity  $2^{69}$  hash operations. Recently, the complexity has further been improved to  $2^{63}$  hash operations [WYY05a].

The code for SHA-0, which is same as (1) but without the rotation, has an even worse minimum weight. The small minimum weight of these codes is an integral part of the attack strategies on these hash functions (see [Wan97a, Wan97b, CJ98, BC04b, BC04a, WYY05b, WYY05c]). The question naturally arises as to why codes with better minimum weight were not employed, even though the coding theory literature [vL98] is rife with codes with proven good minimum weight. However, as we point out later in section 2, none of them comes close to being as efficient to implement in software (i.e., do not have an efficient encoder) as the code (1) above. One is then led to ask if codes more complex than (1), but still easy to implement, could

---

\*This work was done while the author was visiting IBM T.J. Watson Research Center, N.Y.

be shown to have a better minimum distance. Surprisingly, it was not even known how to lower bound the minimum weight of the above SHA-1 code, even though it is related to codes such as Hadamard code [vL98] (we address this relationship in section 2).

The purpose of this paper is three-fold. First, to introduce a novel technique for lower bounding efficient-to-implement codes such as given by (1). Second, to use this technique to lower bound this particular code (which was an open problem). Third, to show how one can design efficient-to-implement codes with a much better minimum distance, and to actually give such a code. We expect our technique to be helpful in designing future practical collision-resistant hash functions.

Before we describe our technique, we mention the specific code we analyze, as this specific example will help in understanding the complexity of the problem and the intricacy of the technique. The code we consider is a  $80 \times 32$  length binary code of dimension  $16 \times 32$ , given by the following recurrence relation (or parity check equations):

$$W_i = \begin{cases} W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15}) \lll 1) & \text{if } 16 \leq i < 36 \\ W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15} \oplus W_{i-20}) \lll 1) & \text{if } 36 \leq i \leq 79 \end{cases} \quad (2)$$

We will show that this code has minimum distance 82, and that too in just the last 64 words (contrast this with SHA-1 which has minimum weight at most 30 [WYY05c], and 192 for the highly inefficient Reed Solomon code described in Section 2). Of course, since the dimension of this code is  $16 \times 32$ , a brute force search of  $2^{16 \times 32}$  is infeasible. Further, it is known that computing minimum weight of an arbitrary linear code is NP-hard (see [Var97]), and that approximating within a constant factor is NP-hard under randomized reduction (see [DMS03]).

Still, there is an additional structure in the above codes (i.e. (1) and (2)), and we intend to exploit that. Note that a codeword of the above codes can be seen as an  $80 \times 32$  matrix, with each column representing the codeword projected on a particular bit position. Further, the above codes have the property that they are closed under column rotations. Such codes are called *quasi-cyclic codes* in the literature, and have been studied extensively (see [TW67, Che92, Lal03, LS05]). As for estimating the minimum weight of such codes by algorithmic means, the presently known techniques are computationally infeasible [Che92, Lal03].

Our novel technique reduces the problem of lower bounding the minimum weight of a  $k \times n$  dimensional quasi-cyclic code to a function of the minimum weight of a few  $k \times n'$  dimensional codes, where  $n'$  is much smaller than  $n$ . We now briefly explain the main idea of our technique, using the above example code given by (2). For any codeword represented as a  $80 \times 32$  matrix, note that either (a) there are no all-zero columns in the codeword, in which case we would like to show that on average there are a few (three, e.g.) non-zero bits in each column, or (b) there is a zero column in the codeword, in which case we would like to show that the code projected on a few columns (say,  $m \ll n$ ) has a large minimum distance.

Unfortunately, there are two major hurdles in this plan (related to case (b)). Consider the first non-zero column next to a zero column (either to the left or the right). It turns out that the code projected on that column is not expected to be any better than the code for SHA-0, and hence we do not expect a minimum weight of more than 15-20 for that column. Thus, we would need  $m$  to be about five to get a minimum weight of 75, in which case the dimension of the projected code is still too large, i.e.  $16 \times 5$ . Further, there are *pathological cases* (and which cannot be avoided) where the code projected on a column yields a minimum weight as low as 1. Thus, we may be forced to consider  $m$  to be much larger than five. The novelty of our approach lies in tackling these two major hurdles. We show that the minimum weight of the sub-code in case (b) can be lower bounded by a function of the minimum weight of a few codes (some of which are subspaces), each of dimension at most  $16 \times 3$ . A “lazy” brute force search with early-stopping then yields a lower bound of 82.

**Other Contributions:** We also use the techniques developed to give a lower bound of 25 (in the last 60 words) on the minimum weight of the codewords of SHA-1 (this was an open problem). A codeword of weight smaller than 25, could potentially lead to an even more drastic attack on SHA-1. As for further advancement of our techniques, we also prove that the minimum weight of our example code (2) is at least 75 when restricted to the last 60 words. Note that front truncations are not equivalent to back truncations for this code.

**Organization:** The rest of the paper is organized as follows: In section 2 we review limitations of known algebraic techniques. In section 3 we give an informal description of the proof technique and the intuition behind why certain codes are easier to analyze. In section 4 we give a detailed proof of a lower bound on the code given by (2). In the Conclusion section, we describe applications of our methods to designing hash functions. In Appendix A, we give detailed algebraic proofs of some lemmas in section 4. In Appendix B, we extend our lower bounds on the code given by (2) to the last 60 and 48 words. In Appendix C, we give a proof of the lower bound on SHA-1 code.

## 2 Limitations of Purely Algebraic Techniques

We first investigate the SHA-0 code restricted to a single column, which is a length 80 binary code of dimension 16, given by the binary parity check equations:

$$a_i = a_{i-3} \oplus a_{i-8} \oplus a_{i-14} \oplus a_{i-16} \quad \text{for } i = 16, \dots, 79 \quad (3)$$

Consider the polynomial  $h(X) = X^{16} + X^{13} + X^8 + X^2 + 1$ , which is known to be a primitive polynomial, as the smallest  $n$  such that  $h(X)$  divides  $X^n - 1$  is  $2^{16} - 1$ . Hence, if the above code was extended up to length  $2^{16} - 1$ , it would be the code generated by the LFSR (Linear Feedback Shift Register) given by primitive polynomial  $h(X)$ . However, such codes are well known [Zie58, KLP68] to be a subcode of first-order Reed Muller codes (also known as Hadamard codes) with one digit dropped. Such codes have an extremely good minimum distance of  $2^{15} - 1$ , or fractional distance  $1/2$ . Unfortunately, nothing useful can be said about this code truncated to just the first 80 bits, based purely on known algebraic methods. In fact, any such code (i.e. using any degree 16 primitive polynomial) has a minimum weight of at most 26, i.e. a fractional distance of less than  $1/3$  (as can be checked by a computer).

The lack of purely algebraic techniques to lower bound even this single column code emphasizes the difficulty of analyzing the more complex codes such as SHA-1 and that given by Equation (2). Of course, if  $h(X)$  above was not primitive, and divided  $X^{80} - 1$ , then we would get a cyclic code of length 80. Such codes can be analyzed much easily, and it is not too difficult to see that the best cyclic code gives a minimum distance of only 8. However, there are non-cyclic linear codes known of minimum distance 31, though they are really difficult to encode. One could also consider cyclic codes of length 85, which have a much better minimum distance and then truncate them. However, the analysis does not extend to codes which do column mixing like SHA-1.

Instead of quasi-cyclic codes as SHA-1 or Equation (2), one could consider cyclic codes of length  $80 \times 32$ , or of an appropriate length. First note that a random code will give minimum distance roughly 475 for a code with rate  $1/4$  and length  $64 \times 32$  (follows from the Gilbert-Varshamov bound). Of course, finding such a good code is infeasible. Alternatively, one can try a Reed Solomon code over  $\mathbb{F}_{2^8}$  of length  $2^8 - 1$  (bytes), and dimension 64 (bytes). Such a code has distance  $256 - 64 = 192$  (over bytes). However, the encoder for this code requires multiplication by various elements in  $\mathbb{F}_{2^8}$ , and is not at all suitable for software implementations. A binary cyclic code of dimension  $16 \times 32$  would also be extremely cumbersome to implement. Similar considerations rule out known good quasi-cyclic codes.

### 3 Intuition behind the code

Let us start by examining why the message expansion code in SHA-1 given by Equation (1) is not satisfactory (observed independently in [RO05] and [MP05]). We can rewrite Equation (1) as follows:

$$\forall i, 0 \leq i \leq 63, \quad W_i = W_{i+2} \oplus W_{i+8} \oplus W_{i+13} \oplus (W_{i+16} \gg\gg 1), \quad (4)$$

where “ $\gg\gg 1$ ” denotes a one bit rotation to the right. The above clearly shows that a difference created in the last 16 words propagates to only up to 4 different bit positions.

One way to remedy this situation is to let  $W_i = (W_{i+2} \gg\gg 1) \oplus W_{i+8} \oplus W_{i+13} \oplus (W_{i+16} \gg\gg 1)$ . Now Equation (1) becomes  $W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-16}) \ll\ll 1 \oplus W_{i-14}$ . Thus, whether you consider the evaluation in the *forward direction* or in the *reverse direction*, the spread of differences to the neighboring columns (i.e. neighboring bits) is more frequent. However, it is not enough to just have a good intuition about the code, but one also needs to prove a good lower bound on the minimum weight of such codes.

The strategy we use to prove lower bounds on such codes is to divide the proof into two main cases. We argue that either there are no zero columns in a codeword (a column in the codeword is the codeword projected on a particular bit position) or starting from an all zero column, the first neighboring non-zero column is actually a codeword in a good code, and so on.

Elaborating on the first case, i.e., when there are no zero columns, if every column has at least 3 bits ON, we are done. So, assume that there is some column which has 1 or 2 bits ON. Thus, there are  $(64 \times 63)/2 + 64$  choices for picking these bits in the column. Having picked these bits, the neighboring column is completely specified by at most 16 bits in that column. Now the two columns together have either weight 6, in which case we are maintaining an average of 3 per column, or the weight of these two columns is at most 5. Thus, our search is quite restricted. We continue in this fashion, noting that the code has to be designed carefully so as to satisfy a property as in Claim 4.6.

As for the second case, we consider a contiguous band of zero columns, bordered on both sides with non-zero columns (we prove that they cannot be same; in fact we prove by a rank argument that there must be at least four consecutive non-zero columns). We have to assure that when a column is zero, and the neighboring column is non-zero (whether to the right or left), the resulting code for the neighboring column is a good code, i.e., with a good minimum weight. Note that this is important since we may possibly have at most 5-6 non-zero columns. Therefore it is desired that the disturbance propagates fast across columns. Unfortunately, this is impossible for the codes we are considering so far.

Consider a SHA-1 like code, with dimension  $16 \times 32$ , and which is invariant under column rotations. Moreover, suppose that the code is of the form

$$W_i = \sum_{t=1}^{16} a_t W_{i-t} + \left( \left( \sum_{t=1}^{16} b_t W_{i-t} \right) \ll\ll 1 \right), \quad (5)$$

where  $a_1, \dots, a_{16}, b_1, \dots, b_{16}$  are boolean. If  $a_{16}$  and  $b_{16}$  are equal, then there is a codeword which is zero everywhere, except for  $W_0$  which is the all 1 32-bit word. Thus for the sake of the argument, assume that  $b_{16} = 0$  and  $a_{16} = 1$ . However in this case, suppose  $t' < 16$  is the largest  $t$  such that  $b_{t'}$  is non-zero. First note that if a column, say  $C^j$ , is zero, then in the column to its right, say  $C^{j-1}$ ,  $C_k^{j-1}$  (for  $k = 0$  to  $15 - t'$ ) can take any value (i.e., are free variables), and the rest of the column  $C^{j-1}$  can be all zero. Further, the propagation to columns  $C^{j-2}$ ,  $C^{j-3}$  etc. can be rather weak.

A similar situation arises when the code is evaluated in the backward direction. The trick is to keep the above free variables few in number, so that the subspace of such *pathological cases* is of a relatively

small dimension. This small dimension is absolutely necessary to keep the exhaustive search over this space tractable. One way to get rid of these pathological free variables is to include a term like  $W_{i-20}$ , as we do in our code. This in fact gets rid of all the pathological variables in the forward direction and thereby yields a fast expansion. In the backward direction at least one pathological free variable per column remains, and we must search over such subspaces.

## 4 A Lower bound on the minimum distance

In this section we will prove a lower bound on the code described in the introduction, and given by Equation (2). As mentioned earlier, this is a general technique for reducing the problem to smaller dimensional codes. However, if the reduction is to codes with dimensions too large, then a brute force search may not be feasible. On the other hand, if the reduction is to codes which have really low minimum weight, then we will not obtain a good bound.

We will see in Claim 4.3 and Claim 4.4 that if the polynomials describing the parity check equations (5) have a certain algebraic property, namely that the polynomial corresponding to coefficients  $a_t$  is irreducible, and does not divide the polynomial corresponding to coefficients  $b_t$ , then some key reduced codes have low dimensions. Although, these are not necessary conditions, they make a good choice. Similarly, if the coefficients  $b_1$  and  $b_{15}$  are both one, then the number of pathological variables per column is small.

We will prove a lower bound on the minimum weight of the code given by Equation (2), but projected on the last 64 words. Clearly, the same bound holds for the full 80 words. The reason we focus on the last 64 words is because the recent attacks on hash functions have shown that the weight of the code in early words (the information words, and a few following words) is mostly immaterial, and hence the weight in the latter words decides the complexity of the attack. Later, in the appendix, we will lower bound the code in the last 60 and 48 words. Note that because of the change in the parity equations at index 36, the codewords restricted to the last 48 words cannot be described as easily as Equation (2).

Since we will be arguing about the weight of this code in the last 64 words, we instead consider the following code  $\mathcal{C}_{64}$ : Let  $W_0, \dots, W_{15}$  be the message blocks. Then

$\mathcal{C}_{64}$  :

for  $i = 16$  to  $63$

$$W_i = \begin{cases} W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15}) \lll 1) & \text{if } 16 \leq i < 20 \\ W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15} \oplus W_{i-20}) \lll 1) & \text{if } 20 \leq i \leq 63 \end{cases} \quad (6)$$

We first prove that this is indeed sufficient. Let  $\mathcal{C}$  be the code defined by Equation (2).

**Lemma 4.1** *If the code  $\mathcal{C}_{64}$  described above has minimum weight at least 82, then  $\mathcal{C}$  has minimum weight at least 82 in its last 64 words.*

*Proof:* Consider any nonzero codeword in  $\mathcal{C}$ , say  $U = \langle U_0, \dots, U_{79} \rangle$ . Denote  $X = \langle U_0, \dots, U_{15} \rangle$  and  $Y = \langle U_{16}, \dots, U_{31} \rangle$  and  $Z = \langle U_{32}, \dots, U_{79} \rangle$ . Therefore  $U = \langle X, Y, Z \rangle$ . From Equation (2) observe that the code  $\mathcal{C}$  is completely determined by specifying any consecutive 16 word block provided the block starts anywhere in 0 to 20, since the rest can then be obtained by solving the recurrence relation. We therefore choose to specify  $Y = \langle U_{16}, \dots, U_{31} \rangle$ , that is we treat  $Y$  as the message symbols. Note that a fixed choice of  $Y$  also fixes  $X$  and  $Z$ . Following this observation it is now clear that  $\langle Y, Z \rangle$  is a codeword in  $\mathcal{C}_{64}$ .

Assume that the minimum weight of  $C64$  is  $d$ . Then we need to show that any non-zero codeword in  $\mathcal{C}$ , has weight at least  $d$  in its last 64 words. This follows provided  $X$  being non-zero implies  $Y$  is non-zero. However,  $Y$  being zero implies  $X$  is zero, as  $X$  is a linear function of  $Y$ . Therefore the minimum weight of  $C64$  is exactly the minimum weight of code  $\mathcal{C}$  in its last 64 words. ■

**Theorem 4.2** *The code  $C64$  as defined by Equation (6) has minimum distance at least 82.*

*Proof:* Let  $d_{\min}$  stand for the minimum weight of the code  $C64$ , and since the code is a linear code it suffices to prove that  $d_{\min} \geq 82$ . From now onwards, we view the codewords of  $C64$  as a matrix that has 32 columns where each column is 64-bit long. It is easy to see that the code is invariant under column rotations. Unless otherwise specified, *the arithmetic in the superscript will be modulo 32*.

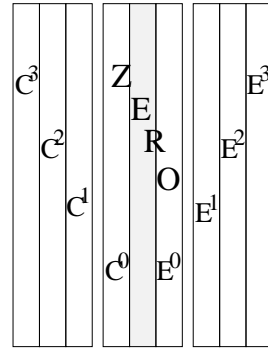
Now consider any non-zero codeword. We break down the proof into two main cases depending upon whether or not a codeword has zero columns.

1. **(All Columns Non-Zero Case:)** Consider any such codeword. Also, consider any non-zero column, w.l.o.g., let it be  $C^0$ . Denote the columns, to the left of it by  $C^1, C^2, \dots, C^{31}$ . Note that all  $C^i$ 's are non-zero. In this case, let  $d_1$  denote the minimum weight of this sub-code (that is the set of codewords that do not have a zero column).

Suppose for any column  $C^j$ , there exists an  $l$ , such that the combined weight of the columns  $C^j, C^{j+1}, \dots, C^{j+l-1}$  is at least  $\mu \times l$ , then we show that  $d_1$  is at least  $(32 - (\ell - 1)) \times \mu + (\ell - 1) = 33 \cdot \mu - \ell(\mu - 1) - 1$ . To see this, we create a partition of the 32 columns into several groups. We pick a non-zero column  $C^j$ . Now by assumption there exists  $\ell$ -columns such that the average weight of each column is at least  $\mu$ . Consider the smallest  $\ell' \leq \ell$  that achieves this. Then put these  $\ell'$  columns  $C^j, C^{j+1}, \dots, C^{j+\ell'-1}$  into a group. Call these columns good columns and the group a good group. We then choose  $C^{j+\ell'}$  and form another group. We continue like this till no more good groups can be created. The remaining columns are then grouped together. Call this group a bad group. Note that the bad group has average weight at least 1. Now let  $e$  be the size of this bad group. Then we have  $(32 - e)$  good columns. Also by assumption,  $e$  could be at most  $\ell - 1$ . Therefore the total weight of the codeword is at least  $d_1 \geq \mu \cdot (32 - e) + e \geq (32 - (\ell - 1)) \times \mu + (\ell - 1) = 33 \cdot \mu - \ell(\mu - 1) - 1$ .

2. **(At Least One Column Zero Case:)** Assume that there is at least one zero column. Let  $d_2$  stand for the minimum weight of this sub-code. W.l.o.g. we can assume that  $C^0$  is a zero column. Further, w.l.o.g. let  $C^0$  be a zero column such that the column to the left of it is non-zero (note that such a column always exists since we are considering a non-zero codeword). Denote the columns to the left of  $C^0$  as  $C^1, C^2, \dots$  (see figure).

Also, going towards the right of  $C^0$ , denote the first non-zero column by  $E^1$  and thereafter  $E^2, E^3, \dots$ . Denote the column to the left of  $E^1$  by  $E^0$ . (Note that it may be possible that  $C^0$  and  $E^0$  are the same column.) We argue that a few columns to the left and right of a band of zero columns must contribute a total weight of at least  $d_{\min}$ .



Also note that w.l.o.g. we can assume that there exists exactly one zero band. To see this observe that if there is more than one distinct zero band, it is always possible to set the non-zero entries between two zero bands to zero (i.e., consider the parity check matrix, if the entries between a zero band is set to zero, it still satisfies the parity check equation). This can only reduce the minimum weight of the codeword. Therefore we can safely assume that there is exactly one zero band.

Next consider  $C^1, C^2, \dots$ . How soon can the sequence yield a zero column, i.e., what is the smallest value of  $j$  such that  $C^j = E^0$ ? In order to answer this question, first note that since  $C^0$  is everywhere zero,  $C^1$  is essentially generated by the code whose parity check equations over  $\mathbb{F}_2$  are given as follows: Denote  $C^1 = \langle y_0, \dots, y_{63} \rangle$ . Then

$$\forall i, 16 \leq i \leq 63, \quad 0 = y_i + y_{i-3} + y_{i-8} + y_{i-14} + y_{i-16}. \quad (7)$$

Similarly for a fixed  $C^1$ , the column  $C^2$  is generated by the code whose parity check equations over  $\mathbb{F}_2$  are given as follows: Denote  $C^2 = \langle x_0, \dots, x_{63} \rangle$ . Then

$$0 = \begin{cases} x_i + x_{i-3} + x_{i-8} + x_{i-14} + x_{i-16} + y_{i-1} + y_{i-2} + y_{i-15} & \text{for } 16 \leq i \leq 19 \\ x_i + x_{i-3} + x_{i-8} + x_{i-14} + x_{i-16} + y_{i-1} + y_{i-2} + y_{i-15} + y_{i-20} & \text{for } 20 \leq i \leq 63 \end{cases} \quad (8)$$

On the other hand  $E^1$  is generated by the code whose parity check equations over  $\mathbb{F}_2$  are given as follows: Denote  $E^1 = \langle w_0, \dots, w_{63} \rangle$ . Then

$$0 = \begin{cases} w_{i-1} + w_{i-2} + w_{i-15} & \text{for } 16 \leq i \leq 19 \\ w_{i-1} + w_{i-2} + w_{i-15} + w_{i-20} & \text{for } 20 \leq i \leq 63 \end{cases} \quad (9)$$

Similarly for a fixed  $E^1$ , the column  $E^2$  is generated by the code whose parity check equations over  $\mathbb{F}_2$  are given as follows: Denote  $E^2 = \langle z_0, \dots, z_{63} \rangle$ . Then

$$0 = \begin{cases} w_i + w_{i-3} + w_{i-8} + w_{i-14} + w_{i-16} + z_{i-1} + z_{i-2} + z_{i-15} & \text{for } 16 \leq i \leq 19 \\ w_i + w_{i-3} + w_{i-8} + w_{i-14} + w_{i-16} + z_{i-1} + z_{i-2} + z_{i-15} + z_{i-20} & \text{for } 20 \leq i \leq 63 \end{cases} \quad (10)$$

We first establish the follow claim.

**Claim 4.3** *If  $C^0$  is zero, and  $C^1$  is non-zero, then  $C^2$  is non-zero.*

*Proof:* Assume otherwise i.e., that  $C^2$  is zero. Consider the  $48 \times 64$  dimensional parity check matrices (essentially Equations (7) and (9)) over  $\mathbb{F}_2$  (see appendix A for matrices  $H_1$  and  $H_2$ ).

Then we need to show that  $H = \begin{pmatrix} H_1 \\ H_2 \end{pmatrix}$  has full rank. To do that it is enough to show that there are 64 linearly independent rows. We consider the 48 rows of  $H_1$  and 16 additional rows, namely 5<sup>th</sup> through 20<sup>th</sup> rows of  $H_2$ . We reduce the problem to showing that a certain equation over polynomial ring  $\mathbb{F}_2[x]$  does not have solutions in a restricted set of polynomials. We associate with the vector  $c = \langle c_0, \dots, c_{63} \rangle$  in  $\mathbb{F}_2^{64}$  the polynomial  $c(x) = \sum_{i=0}^{63} c_i x^i$  in  $\mathbb{F}_2[x]$ . Then the following polynomials can be associated with the 1<sup>st</sup> and 5<sup>th</sup> rows of matrix  $H_1$  and  $H_2$ , respectively:

$$p(x) \stackrel{def}{=} x^{16} + x^{13} + x^8 + x^2 + 1,$$

$$r(x) \stackrel{def}{=} x^{19} + x^{18} + x^5 + 1.$$

Further note that the  $i^{th}$  (note  $1 \leq i \leq 48$ ) row of  $H_1$  then gets associated with  $x^{i-1}p(s)$ . Similarly the  $j^{th}$  (note we restrict ourselves to  $5 \leq j \leq 20$ ) row of  $H_2$  then gets associated with  $x^{j-5}r(s)$ . Therefore, observe that if the 80 rows that we are considering were dependent then we can translate that to a non-zero solution of the following polynomial equation:

$$p(x)\alpha(x) + \beta(x)r(x) = 0,$$

with additional constraints that  $\text{degree}(\alpha) \leq 47$  and  $\text{degree}(\beta) \leq 15$ . However, it is well known that  $p(x)$  is irreducible, therefore if such an equation holds then it must be the case that  $p(x)$  divides  $r(x)$ . However, it is easy to check that  $p(x)$  does not divide  $r(x)$ , thus leading to a contradiction.

Therefore  $H$  has full rank. ■

We now strengthen the claim slightly.

**Claim 4.4** *If  $C^0$  is zero, and  $C^1$  is non-zero, then  $C^2, C^3$  is non-zero.*

*Proof:* Consider the following polynomials :

$$p(x) \stackrel{def}{=} x^{16} + x^{13} + x^8 + x^2 + 1,$$

$$q(x) \stackrel{def}{=} x^{15} + x^{14} + x,$$

$$r(x) \stackrel{def}{=} x^{19} + x^{18} + x^5 + 1 = x^4 \cdot q(x) + 1.$$

Let  $H_1$  and  $H_2$  be as above.

First of all note that  $H_2$  has full rank. (This is clear from the matrix. Otherwise, note that we could have an identity

$$q(x) \cdot a(x) + r(x) \cdot b(x) = 0$$

with  $\text{degree}(a) \leq 3$  and  $\text{degree}(b) \leq 43$ . Since  $\text{degree}(q \cdot a) < \text{degree}(r)$ , this cannot happen.) Now we will show that the rank of the matrix

$$\begin{pmatrix} H_2 & 0 \\ H_1 & H_2 \\ 0 & H_1 \end{pmatrix}$$

is at least 128. Since  $H_1$  has full rank, observe that

$$\begin{pmatrix} H_1 & H_2 \\ 0 & H_1 \end{pmatrix}$$

has rank at least 96. So consider the following 92 independent rows from the above matrix, namely  $5^{th}$  row onwards. We also argue that another additional  $5^{th}$  through  $40^{th}$  rows of the top  $H_2$  are also independent. If not, then they would satisfy the following polynomial equations

$$\begin{array}{l} \alpha(x)p(x) + \beta(x)r(x) = 0 \quad (11) \\ x^4\beta(x)p(x) + \gamma(x)r(x) = 0 \quad (12) \end{array} \left| \begin{array}{l} \text{with restrictions} \\ \text{degree}(\alpha) \leq 47, \\ \text{degree}(\beta) \leq 43, \text{ and} \\ \text{degree}(\gamma) \leq 35. \end{array} \right.$$

Since  $p(x)$  is an irreducible polynomial, and  $p(x) \nmid r(x)$ , observe from Equation (11) that  $p(x) \mid \beta(x)$ . Hence, set  $\beta(x) = \mu(x)p(x)$ . Substituting in Equation (12) we get

$$x^4p(x)^2\mu(x) + \gamma(x)r(x) = 0.$$

Since  $p(x)$  is irreducible, and  $p(x) \nmid r(x)$ , and  $x \nmid r(x)$ , it must hold that  $x^4 p(x)^2 \mid \gamma(x)$ . But that is impossible, since  $\text{degree}(\gamma) \leq 35 < 36 = \text{degree}(x^4 p(x)^2)$ . ■

The above proof also highlights that for the rank to be full the recurrence relation must satisfy nice properties. In fact, the following claim strengthens it further.

**Claim 4.5** *If  $C^0$  is everywhere zero, and  $C^1$  is non-zero, then so is  $C^2, C^3$  and  $C^4$ .*

*Proof:* This claim is proven by checking that the system of equations involved have a full rank, which can be checked by a computer. An algebraic proof of this claim is extremely difficult, although for a random quasi-cyclic code of the kind we are considering, it can be shown that with high probability the rank of the system of equations is close to  $64 \times 3$  (proof deferred to the full version of the paper).

In fact the above lemma can further be generalized as follows: Consider the subcode (i.e., set of codewords) where there are at most  $s$  ( $5 \geq s \geq 4$ ) consecutive non-zero fixed columns and the rest are filled with zero, then this subcode has dimension exactly  $16 \cdot s - 48$ . This can be verified by a computer just as for Claim 4.5. Then we choose an  $s_1$  so that a search in a space of dimension  $16 \cdot s_1 - 48$  is feasible. Thus, we choose  $s_1 = 5$ . Let  $d_{21}$  denote the minimum weight in this subcode, i.e the sub-code which has at most  $s_1$  consecutive non-zero columns (and the rest are all zero).

Now we restrict our attention to the cases where there are at least  $s_2 \stackrel{\text{def}}{=} s_1 + 1$  consecutive non-zero columns.

- (a) **At least  $s_2$  consecutive columns:** Let  $s_f = \lceil \frac{s_2}{2} \rceil$  and consider columns  $C^1, C^2, \dots, C^{s_f}$ . Let  $d_f$  be the minimum of the combined weight of these columns. (We would like  $d_f = 82/2 + \epsilon$ .) In particular, we also need to make sure that the dimension of this space (which is at most  $16 \times s_f$ ) is small.

Once we do this, we next consider  $s_b = \lfloor \frac{s_2}{2} \rfloor$  columns  $E^1, \dots, E^{s_b}$ . Notice that the dimension of this space is small ( $16 \times s_b \leq 16 \times s_f$ ) and hence searchable. However, it turns out that the minimum weight can be extremely small. Fortunately, all these bad cases can be characterized into what we call *pathological cases*. Recall Equation (9), the constraints induced on  $E^1$ . A quick observation reveals that its free variables are the first 15 bits and the very last bit. If the values taken by  $E^1$ 's first 15 bits are zero, then we call it a pathological case, and non-pathological otherwise.

- i. (**Non-Pathological Case:** i.e., Not all of the first 15 bits of  $E^1$  are zero.) Assuming  $E^1$  to be non-pathological, and let  $d_b$  be the minimum of the combined weight of the columns  $E^1, \dots, E^{s_b}$ . (We would like  $d_b = 82/2 - \epsilon$ .) Also, note that by the assumption the columns  $C^1, \dots, C^{s_f}$  and  $E^1, \dots, E^{s_b}$  are all distinct. Thus in the non-pathological case we see any codeword must have weight at least  $d_3 = d_f + d_b$ .
- ii. (**Pathological Case:**) Therefore only the pathological cases remains. This is the most subtle and difficult case. Going back to Equation (9), we note that in this case it must hold that  $w_{63} = 1$  and for all  $0 \leq i \leq 62, w_i = 0$ . We call such  $w$  *pathological*. Now consider Equation (10). We can have two cases here.

In the first case, assume that the first 15 variables of  $z$  are zero. In that case, it must hold that  $z_{62} = 1$ . (Plugging in  $i = 16$  to  $62$  in Equation (10) will yield  $z_j = 0$  for all  $15 \leq j \leq 61$  since  $w_i = 0$  for these values.) Also note that  $z_{63}$  is free. In this case, we also call  $z$  pathological. In fact this may continue along the diagonal i.e.,  $E^3, E^4, \dots$  may be pathological. If that happens then it is easy to show that the first non-zero bits of  $E^3$  will be its 61<sup>st</sup> bit, that of  $E^4$  will be 60<sup>th</sup> bit and so on. Also each column will have a free variable in its 63<sup>rd</sup> bit.

In the second case, we assume that not all of its first 15 variables are zero. We call such  $z$ 's to be non-pathological.

Now in this pathological case we need to consider more columns. Firstly note that it can never be the case that only pathological columns are the non-zero columns. (Otherwise  $C^1$  will be pathological, a contradiction.) In fact it can be argued that there has to be at least 4 non-pathological columns (similar to Claim 4.5). Thus this sets an upper bound, say  $p_{\max}$ , on the number of pathological columns. Thus if we assume there are  $p$  pathological columns  $E^1, E^2, \dots, E^p$  and then  $n$  non-pathological columns  $E^{p+1}, \dots, E^{p+n}$ . Now note that two cases can arise. Either all columns  $C^1, \dots, C^{s_f}$  and  $E^1, E^2, \dots, E^{p+n}$  are distinct or  $C^{s_f} = E^{p+n'}$  for some  $n' \leq n$ .

**Case A:** In the first case, note that the dimension of the search space is  $16 \times n + p$ . The main idea here is to choose  $n$  and  $p$  appropriately so that the space remains searchable, that is if we increase  $p$  by too many (say 16) then we should decrease  $n$  (say by 1). If the minimum of the combined weight of the  $p$  many pathological and  $n$  many non-pathological columns is  $d_{bp}$  (we would like it to be  $82/2 - \epsilon$ ), then the combined minimum weight in this subcode is at least  $d_{pA} = d_f + d_{bp}$ . In general, we can consider  $p_1, p_2, \dots, p_l = p_{\max}$  many pathological columns with  $n_i = s_b - \gamma_i$  for  $i = 1, \dots, l$ , many non pathological columns, where  $\gamma_i$ s are proportionately small (so that search space remains tractable), and if the minimum of the combined weight of these columns is  $d_{bpA_i}$  (we would like it to be  $82/2 - \epsilon$ ), then the minimum of the combined weight of the corresponding subcode is at least  $d_{pA_i} = d_f + d_{bpA_i}$ .

**Case B:** In the second case, define  $n$  be the smallest  $n'$  such that  $C^{s_f} = E^{p+n'}$ . Now consider the subcode where exactly  $C^1, C^2, \dots, C^{s_f} = E^{p+n}, E^{p+n-1}, \dots, E^1$  columns are non-zero (i.e., fix a set of columns). Then (see Appendix A Claim A.1) the nullity of the system can be shown to be

$$p + 64 \times (s_f + n - 1) - 48 \times (s_f + n) = p + 16 \cdot s_f + 16 \cdot n - 64.$$

We employ similar idea as in the previous case. We consider  $p_1, p_2, \dots, p_l = p_{\max}$  many pathological columns along with corresponding  $n_i = (s_f + s_b - 1) - \gamma_i$  for  $i = 1, \dots, l$  many non pathological columns (note that  $C^1, C^2$  and  $C^3$  are non pathological and included in this calculation), and let the minimum of the combined weight of these columns be  $d_{pB_i}$ . (We would like  $d_{pB_i} \geq 82$  for each  $i = 1, \dots, p_{\max}$ .)

Note that this exhausts all possibilities. Thus  $d_2 \geq \min\{d_{21}, d_3, d_{pA_i}, d_{pB_i} | i \in \{1, \dots, l\}\}$ , and  $d_{\min} \geq \min\{d_1, d_2\}$ .

We choose the parameters carefully and do an exhaustive search. We record the following claim.

**Claim 4.6** *For any non-zero column  $C^j$ , there exists  $k, 0 \leq k \leq 7$  such that the combined weight of columns  $C^j, C^{j+1}, \dots, C^{j+k}$  is at least  $3 \cdot (k + 1)$ .*

*Proof:* This is easily verified by a computer program. We mention that for  $k \leq 6$ , an average of 3 cannot be assured (see Appendix D for an example). ■

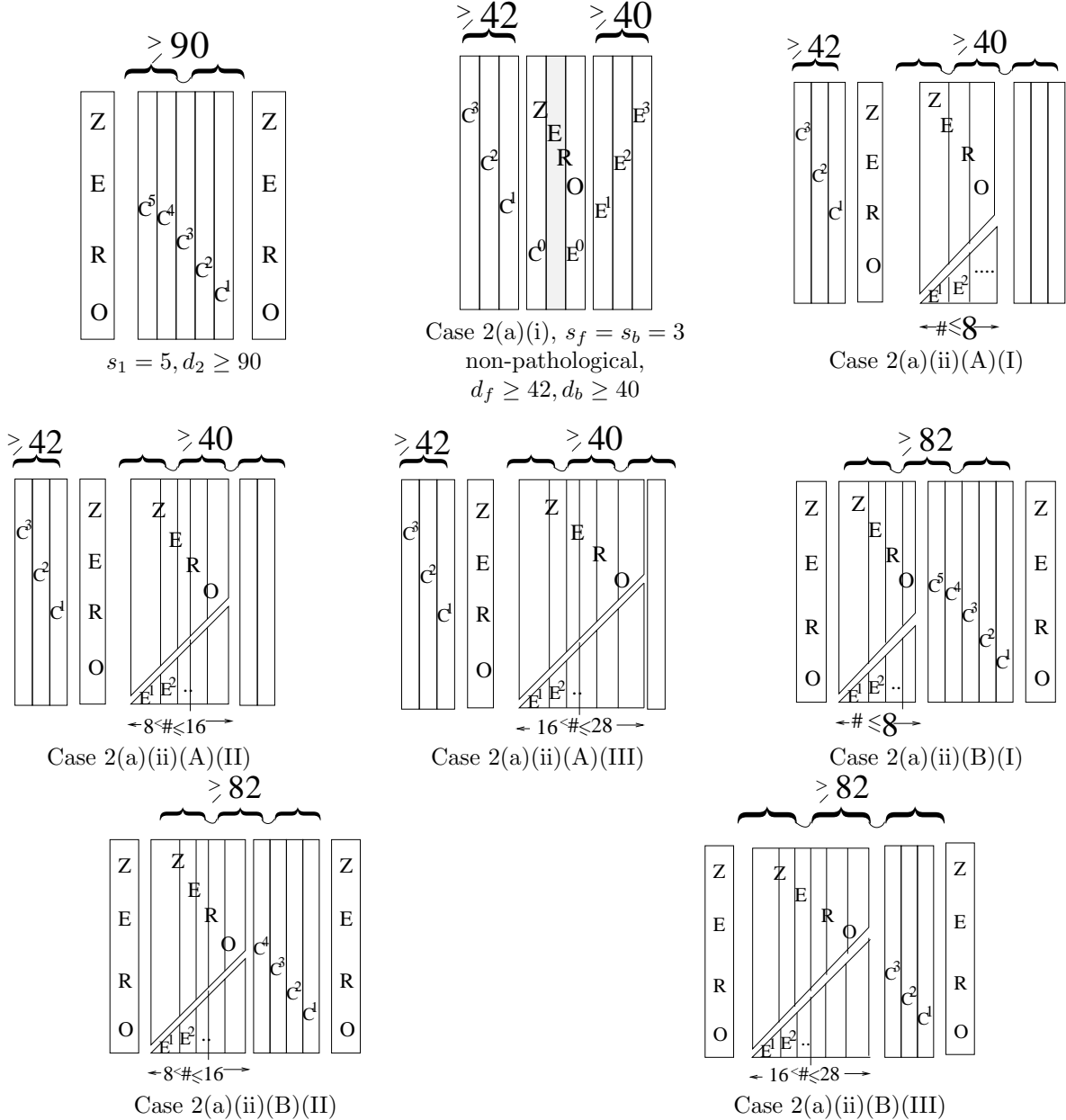
With this we see that with  $\ell = 8$  and  $\mu = 3$  we get  $d_1 = 82$ .

We set  $s_1 = 5$  and get  $d_{21} = 90$  by an exhaustive search. Note that it is important to choose  $s_1$  small so as to keep the space searchable.

Thus  $s_f = 3 = s_b$  and by an exhaustive search, we get  $d_f = 42$  (i.e.,  $\epsilon = 1$ ) and  $d_b = 40$ . This implies  $d_3 \geq 82$ . Recall that this is the non-pathological case, and hence reasonable expansion can be expected.

For pathological cases, we first  $p = 8$  and  $n = 3$ . We get  $d_{bp1} \geq 40$ . Similarly, setting  $p = 16$  and  $n = 2$ , we get  $d_{bp2} \geq 40$  and setting  $p = 28$  and  $n = 1$  we get  $d_{bp3} \geq 40$ . Note in these cases the columns  $E^{p+n}$  and  $C^3$  are distinct.

Similarly, for the second subcase of pathological cases, setting  $p = 8, n = 3$  we get  $d_5 = d_{5,1} \geq 82$ ; setting  $p = 16, n = 2$  we get  $d_{5,2} \geq 82$ , and setting  $p = 28, n = 1$  we get  $d_{5,3} \geq 82$ . Note that in this case  $\exists n'$  s.t.  $E^{p+n'} = C^3$ . Also note that choosing  $n = n'$  is sufficient. This completes the proof. ■



Various Cases in the proof of Theorem 4.2

(weights referred to the combined weights of the columns)

We remark that the minimum weight of this code can at most be 82 and therefore our result is tight (see Appendix D). Extending our approach, we can further prove the following theorem whose proof has been relegated to appendix (see Appendix B).

**Theorem 4.7** *The code C64, as defined by Equation (6), has minimum weight at least 75 (and at least 52) in its last 60 words (and in its last 48 words, respectively).*

We remark that a simple variants of the above technique can be used to give a lower bound on the minimum weight of SHA-1 (of course, there are much fewer cases to consider here). Specifically we have the following theorem.

**Theorem 4.8** *SHA-1 message expansion code has minimum weight 25 in the last 60 words.*

*Proof:* See appendix C.

## 5 Conclusion

In this paper we have shown how lower bounds on minimum weight of quasi-cyclic linear codes of dimension  $m \times n$  given by parity equations of the form

$$W_i = \sum_{t=1}^i a_{it}W_{i-t} + \left( \left( \sum_{t=1}^i b_{it}W_{i-t} \right) \lll 1 \right) \quad \text{for } i \geq n,$$

can be obtained by reducing the problem to the minimum weight of significantly smaller dimensional codes. Note that this equation is more general than Equation (5), and Equation (2) is of this form rather than the simpler Equation (5). In some cases, we obtain the exact minimum weight, including the example codes we considered. An obvious generalization is to consider three or more column mixing (the equation above has only two column mixing), which could lead to codes with even better minimum distance.

A common paradigm for designing hash functions, including MD5[Riv92], SHA-0, SHA-1 and SHA-2[Uni02] is the following: the 512-bit message is first expanded into  $N$  words, and then the  $N$  words are used as step keys (sometimes known as round keys) in  $N$  steps of a (non-linear) block cipher invoked on an initial vector. The output of the block cipher is the output of the compression function. As pointed out in the Introduction, one of the key ingredients of the recent differential attacks on MD5, SHA-0, and SHA-1 has been their poor message expansion (in terms of minimum weight) into the  $N$  words. Also, it is not known how to lower bound the SHA-2 message expansion. Thus, we consider our novel technique to be an important advance in the design of collision-resistant hash functions.

Finally, we remark that the code given by Equation (2) can be encoded in software with at most a 5% overhead over the SHA-1 code. In [JP05] we analyze the security of a hash function, which is exactly the same as SHA-1 except for replacing its message expansion by Equation (2), and show that it is resistant to recent differential attacks. There we address the relationship between minimum weight and differential attacks in more details.

## 5.1 Acknowledgment

We thank W. Eric Hall, Ronald Rivest, Pankaj Rohatgi, Patrick Solé and Alexandar Vardy for helpful comments.

## References

- [BC04a] E. Biham and R. Chen. Near collisions of SHA-0. In *Crypto, Lecture Notes in Computer Science 3152*, 2004.
- [BC04b] E. Biham and R. Chen. New results on SHA-0 and SHA-1. In *Short talk presented at CRYPTO'04 Rump Session*, 2004.
- [Che92] V. V. Chepyzhov. New lower bounds for minimum distance of linear quasi-cyclic and almost linear cyclic codes. In *Problems of information Transmission, Vol. 28, No. 1*, 1992.
- [CJ98] F. Chabaud and A. Joux. Differential collisions in SHA-0. In *Crypto, Lecture Notes in Computer Science 1462*, 1998.
- [DMS03] I. Dumer, D. Micciancio, and M. Sudan. Hardness of approximating the minimum distance of a linear code. In *IEEE Transaction on Information Theory, 49(1)*, 2003.
- [JP05] Charanjit S. Jutla and Anindya C. Patthak. Is SHA-1 conceptually sound? Cryptology ePrint Archive, Report 2005/350, 2005. <http://eprint.iacr.org/>.
- [KLP68] T. Kasami, S. Lin, and W. W. Peterson. New Generalization of the Reed-Muller Codes Part I: Primitive Codes. . *IEEE Transactions on Information Theory*, IT-14(2):189–199, March 1968.
- [Lal03] K. Lally. Quasicyclic codes of index  $\ell$  over  $\mathbb{F}_q$  Viewed as  $\mathbb{F}_q[x]$ -submodules of  $\mathbb{F}_{q^t}[x]/\langle x^m - 1 \rangle$ . In *Lecture Notes in Computer Science, Vol. 2643, Springer*, 2003.
- [LS05] S. Ling and P. Solé. Structure of quasi-cyclic codes III: Generator theory. In *IEEE Transaction on Information Theory*, 2005.
- [MP05] K. Matusiewicz and J. Pieprzyk. Finding good differential patterns for attacks on SHA-1. In *International Workshop on Coding and Cryptography*, 2005.
- [Riv92] R. Rivest. RFC1321: The MD5 message-digest algorithm. In *Internet Activities Board*, 1992.
- [RO05] V. Rijmen and E. Oswald. Update on SHA-1. In *Lecture Notes in Computer Science, Vol. 3376, Springer*, 2005.
- [TW67] R. L. Townsend and E. J. Weldon. Self-orthogonal quasi-cyclic codes. In *IEEE Transaction on Information Theory*, 1967.
- [Uni02] United States Department of Commerce, National Institute of Standards and Technology, Federal Information Processing Standard Publication #180-2. *Secure Hash Standard*, August, 2002.
- [Var97] A. Vardy. The intractability of computing the minimum distance of a code. In *IEEE Transaction on Information Theory, 43(6)*, 1997.
- [vL98] J. H. van Lint. *Introduction to Coding Theory*. Springer, 1998.

- [Wan97a] X. Y. Wang. The collision attack on SHA-0. In Chinese, 1997.
- [Wan97b] X. Y. Wang. The Improved collision attack on SHA-0. In Chinese, 1997. <http://www.infosec.edu.cn/>.
- [WYY05a] X. Wang, A. Yao, and F. Yao. New collision search for SHA-1. In *Short talk presented at CRYPTO'05 Rump Session*, 2005.
- [WYY05b] X. Wang, H. Yu, and Y. L. Yin. Efficient collision search attacks in SHA-0. In *Crypto, Lecture Notes in Computer Science 3621*, 2005.
- [WYY05c] X. Wang, H. Yu, and Y. L. Yin. Finding collisions in the full SHA-1. In *Crypto, Lecture Notes in Computer Science 3621*, 2005.
- [Zie58] N. Zierler. On a variation of the first-order reed-muller codes. In *M.I.T. Lincoln Lab., Group Report, 34-80, Lexington, Mass.*, October 1958.

## A Rank proofs

$$\begin{pmatrix} 1010000010000100100000 & \cdots & 00000000000000000000 \\ 0101000001000010010000 & \cdots & 00000000000000000000 \\ & \ddots & \\ 0000000000000000000000 & \cdots & 010100000100001001 \end{pmatrix}$$

$H_1$

$$\begin{pmatrix} 0100000000000011000000 & \cdots & 00000000000000000000 \\ 0010000000000001100000 & \cdots & 00000000000000000000 \\ 0001000000000001100000 & \cdots & 00000000000000000000 \\ 0000100000000000110000 & \cdots & 00000000000000000000 \\ 1000010000000000011000 & \cdots & 00000000000000000000 \\ 0100001000000000001100 & \cdots & 00000000000000000000 \\ & \ddots & \\ 0000000000000000000000 & \cdots & 100001000000000000110 \end{pmatrix}$$

$H_2$

Recall that we used  $E^0$  to denote a column that is zero everywhere. Also, recall that the columns left to  $E^0$  are denoted  $E^1, E^2$  and so on. In the following claim, we will assume  $3 \leq n$ .

**Claim A.1** *Let  $E^1, E^2, \dots, E^p$  be  $p$  pathological columns. Also, let  $E^{p+1}, E^{p+2}, \dots, E^{p+n}$  be  $n$  non-pathological columns. Further assume that  $E^{p+n+1} = C^0$  is everywhere zero. If the nullity of the parity check equations resulting from these columns with  $p = 0$  is  $16 \cdot n - 48$ , then the nullity of the parity check equations resulting from these columns with any  $p \leq 28$  is*

$$p + 16 \cdot n - 48.$$



Thus the nullity of the system is

$$64 \cdot n + \frac{p(p+1)}{2} - \left( 48(n+1) + \frac{p(p-1)}{2} \right) = p + 16 \cdot n - 48.$$

This completes the proof. ■

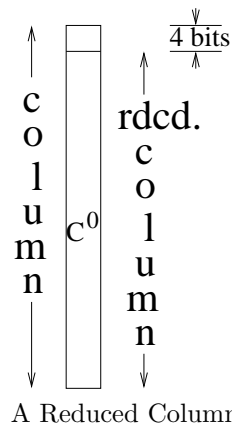
## B The Last Sixty Words

In this section, we prove that the minimum weight of the code  $\mathcal{C}$  in the last 60 words is at least 75 and that in the last 48 words is at least 52, respectively. In general, our proof strategy is robust, i.e., it can in principle be adapted to estimate the minimum weight of this code in the last  $4 \cdot n$  (where  $n$  is an integer) number of steps, though the dimension of the search space increases by an additive factor of  $(64 - 4 \cdot n)$  and may make it computationally infeasible. On the other hand, when  $n$  gets smaller, say  $n \leq 12$ , we may only need to show an average 2 per column viz a viz Claim 4.6. Since most of our search is conducted using early-stopping, the large dimension is not expected to be a problem.

Next, observe that the minimum weight of the code  $\mathcal{C}_{64}$  in the last 60 words yields a lower bound on the minimum weight of the code  $\mathcal{C}$  in the last 60 words. Reviewing the proof of Theorem 4.2, it may be observed that in case 2 (i.e., **At Least One Column Zero Case**) we either consider a codeword (case 2(b)(ii)(A)(II), case 2(b)(ii)(B)(II) and case 2(b)(ii)(C)(II)) or consider few columns (in the remaining cases) which can always be extended to get a valid codeword. Therefore in these cases just counting the weight of the last 60 words gives a lower bound on the minimum weight of the code in the last 60 words. However, the same is not true for case 1 (i.e., **All Columns Non-zero Case**). We handle this case carefully. This then allows us to prove the following theorem.

**Theorem B.1** *The code  $\mathcal{C}_{64}$ , as defined by Equation (6), has minimum weight at least 75 in its last 60 words.*

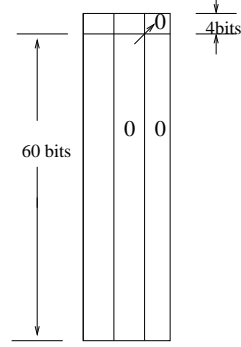
*Proof:* Consider any column of length 64 bits. A column restricted to its bottom most 60 bits will henceforth be referred to as a *reduced column* (see figure).



Unless otherwise mentioned, we will use the same name, eg.,  $C^0$ , to denote a column and its reduced column. We divide the proof into three main cases.

1. (**All Columns Are Non-zero But Reduced Column Can Be Zero Case**): Consider any such codeword. Also consider any non-zero column, w.l.o.g., let it be  $C^0$ . Denote the columns, to the left

of  $C^0$  by  $C^1, C^2, \dots, C^{31}$ . Note that by assumption all columns are non-zero. Then observe that due to this assumption no two consecutive reduced columns can be zero everywhere. To see this let  $C^0$  and  $C^1$  be the columns such that their reduced columns are everywhere zero. Let  $C^1$  be the column left to  $C^0$ . Denote  $C^0$  by  $x = \langle x_0, x_1, \dots, x_{63} \rangle$  and  $C^1$  by  $y = \langle y_0, y_1, \dots, y_{63} \rangle$ . Note that by the assumption  $x_i = y_i = 0$  for all  $i = 4, \dots, 63$ . Now consider the parity check equations of  $C^{64}$  and set  $i = 20$ .



We get

$$y_{20} + y_{17} + y_{12} + y_6 + y_4 + x_{19} + x_{18} + x_5 + x_0 = 0,$$

which implies  $x_0 = 0$ . Similarly by setting  $i = 21, 22, 23$ , it can be seen that  $x$  is everywhere zero.

We can therefore safely assume that no two consecutive reduced columns are zero. Then, the following can be easily verified by a computer program.

**Claim B.2** For any non-zero column  $C^i$ , there exists  $k, 0 \leq k \leq 7$  such that the combined weight of the reduced columns  $C^i, C^{i+1}, \dots, C^{i+k}$  is at least  $3 \cdot (k + 1)$ .

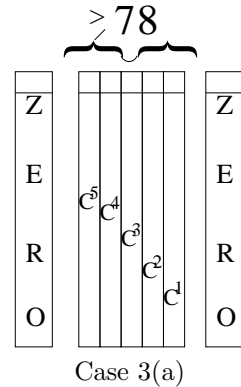
Note that although we restrict ourselves to at most 2 bits ON in reduced  $C^0$ , we must consider all 16 possibilities for the first 4 bits of  $C^0$  to be able to define reduced column  $C^1$  (from 16 bits in reduced column in  $C^1$  and all the bits in  $C^0$ ). Despite this the search is easily conducted.

Then, following the same line of argument as in Case 1 (**All Columns Non-Zero Case**) of Theorem 4.2, it can be shown that the total weight of the reduced columns is at least 78. This is because 25 columns yield at least 75 and the remaining seven columns yield at least 3 (since two consecutive reduced columns contribute at least 1).

2. (**At Least One Column Zero Case**): This case can be handled as the **Zero Case** in the proof of theorem 4.2. We consider the same number of cases and we count only the last 60 bits in a column. We skip the details and summarize below the results we obtain.

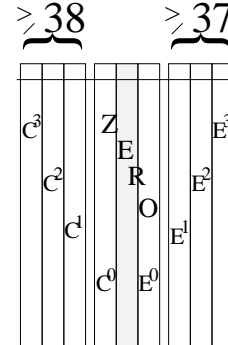
(a) **Number Of Consecutive Non-Zero Columns Is At Most Five:**

The combined weight of the 5 non-zero column is then at least 78.



(b) **Number Of Consecutive Non-Zero Columns Is At Least Six:** The combined weight of three reduced columns to the left of a zero band is at least 38.

i. **(Non-Pathological Case)** The combined weight of three reduced columns to the right of a zero band is at least 38.



Case 3(b)(i)

Therefore the combined weight of three reduced columns to the left of a zero column and that of three reduced columns to the right of a zero column yields (assuming they are distinct) at least 75.

ii. **(Pathological Case)**

A. **# of Pathological columns  $\leq 8$**

(I). **6<sup>th</sup> and earlier non-pathological columns are non-zero** : The combined weight of the pathological reduced columns and the first three non-pathological reduced columns to the right of the pathological columns is at least 37.

(II). **6<sup>th</sup> or earlier non-pathological column is zero**: The combined minimum weight of these reduced columns is at least 75.

B. **8 < # of Pathological columns  $\leq 16$**

(I). **5<sup>th</sup> and earlier non-pathological columns are non-zero** : The combined weight of the pathological reduced columns and the first two non-pathological reduced columns to the right of the pathological columns is at least 37.

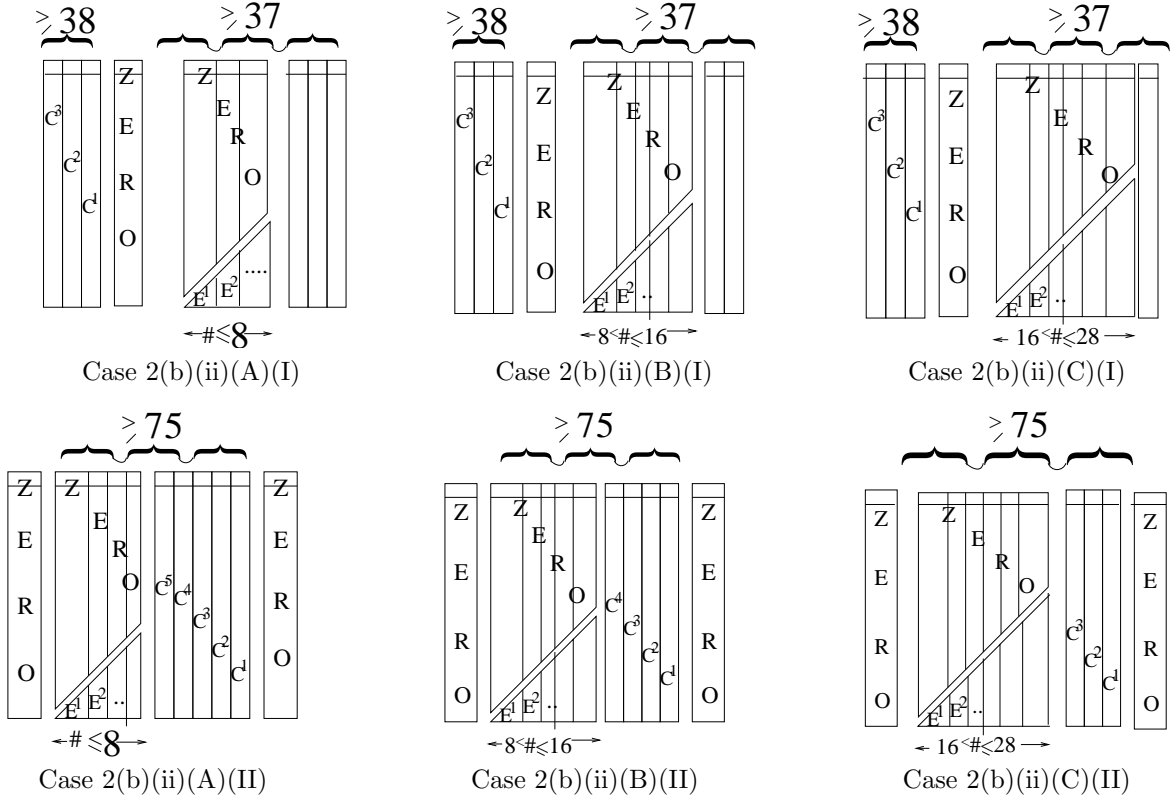
(II). **5<sup>th</sup> or earlier non-pathological column is zero**: The combined minimum weight of these reduced columns is at least 75.

C. **16 < # of Pathological columns  $\leq 28$**

(I). **4<sup>th</sup> and earlier non-pathological columns are non-zero** : The combined weight of the pathological reduced columns and the first non-pathological reduced columns to the right of the pathological columns is at least 37.

(II). **4<sup>th</sup> or earlier non-pathological column is zero**: The combined minimum weight of these reduced columns is at least 75.

Therefore, in all these cases the combined weight of the reduced column is at least 75. This establishes the theorem. ■



Various Cases in the proof of Theorem B.1  
 (weights referred to the combined weights of the reduced columns)

Note that our result is tight. The codeword we cite in the previous subsection achieves this bound.

## B.1 The Last Forty-Eight Words

In this subsection, we prove that the code  $C_{64}$  has minimum weight at least 52 in its last 48 words. As mentioned previously, this proof is more computation intensive as the dimension of the search space increases by an additive factor of 16. The good thing is that we need to show an average 2 per column, viz a viz Claim 4.6. This makes our search, conducted using early-stopping, feasible in spite of the apparent large dimension.

It is easy to observe that the minimum weight of the code  $C_{64}$  in the last 48 words yields a lower bound on the minimum weight of the code  $\mathcal{C}$  in the last 48 words. The proof uses the same technique as in the proof of Theorem B.1. Recall that in that proof (that is the proof of Theorem B.1) there are cases where we either consider a codeword or consider few columns which can always be extended to get a valid codeword. In those cases, just counting the weight of the last 48 words suffices to give a lower bound on the minimum weight of the code in the last 48 words. In the remaining case, mimicking the proof of Theorem B.1, we consider reduced columns (here restricted to last 48 entries). We then can verify that under the assumption



## C Proof of Theorem 4.8

*Proof:* First observe that it suffices to consider the code of length 60 given by the recurrence relation

$$\text{for } i = 16 \text{ to } 59 \quad W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1.$$

We view each codeword as a matrix consisting of 32 columns, each of length 60. Note that the code is invariant under column rotations.

Now if a codeword has all columns non-zero, we are done, as that gives minimum weight at least 32. So, assume that the codeword has one or more zero columns and at least one non-zero column.

Let the column  $C^1$  be the first non-zero column to the right of a band of zero columns. Let the column  $C^1$  be represented by the vector  $\langle x_i \rangle_{i=0}^{59}$ . Then  $x$  satisfies

$$\text{for } i = 16 \text{ to } 59 \quad x_{i-3} \oplus x_{i-8} \oplus x_{i-14} \oplus x_{i-16} = 0,$$

which can be rewritten as :

$$\text{for } i = 13 \text{ to } 56 \quad x_i \oplus x_{i-5} \oplus x_{i-11} \oplus x_{i-13} = 0. \tag{13}$$

Thus for any choice of the first 13 bits of  $x$  (i.e.,  $i = 0$  to 12), the bits from  $i = 13$  to 56 are determined by the above recurrence. The bits  $x_{57}$ ,  $x_{58}$  and  $x_{59}$  are independent, and can be chosen independently.

Similarly, let  $C^2$  be the column to the right of  $C^1$ , and let the column be denoted by vector  $y$ . Then,

$$\text{for } i = 16 \text{ to } 59 \quad y_{i-3} \oplus y_{i-8} \oplus y_{i-14} \oplus y_{i-16} = x_i,$$

which can be rewritten as

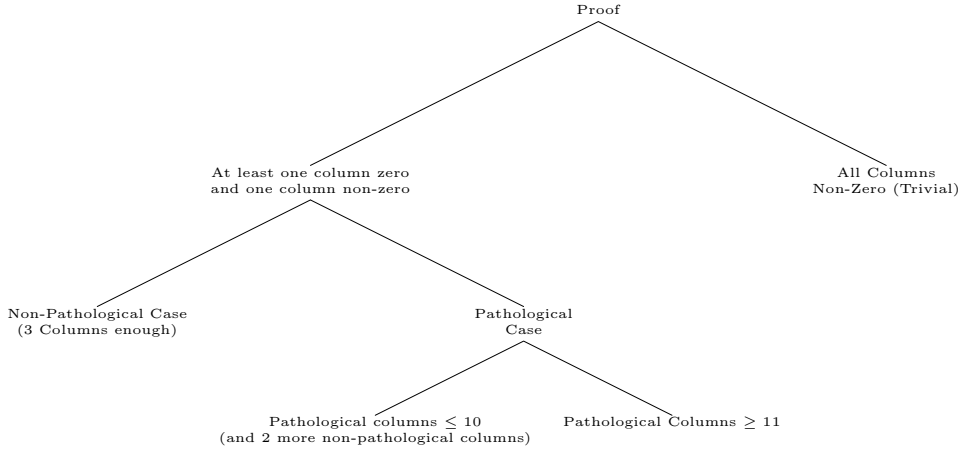
$$\text{for } i = 13 \text{ to } 56 \quad y_i \oplus y_{i-5} \oplus y_{i-11} \oplus y_{i-13} = x_{i+3}. \tag{14}$$

Again, given the full vector  $x$ , and the first 13 bits of  $y$ , the remaining bits of  $y$  are given by this relation (except the last three bits, which remain independent). We continue like this to the next column  $C^3$ , with  $z$  denoting the vector. We mention that if the first 13 bits of  $x$  are non zero, then the code expands fast, that is the individual weight of  $x$  and  $y$  are reasonably good.

So, ideally, we would like to show that no matter how one chooses those bits in  $x$ , and in  $y$ , and in  $z$ , the total weight in the three columns is at least 25. (Of course, we stop early, if just two columns sufficed.) However this is not always true, as  $C^1$  which is required to be the first non-zero column could be *pathological* in the sense that its first 13 bits can be all zero, and hence the bits from  $i = 13$  to 56 can also be all zero, and the only non-zero entries come from  $x_{57}$ ,  $x_{58}$  or  $x_{59}$ . We call such a column pathological. Similarly, given that  $C^1$  is pathological,  $C^2$  can also be pathological, with non-zero entries in only its last 6 entries this time, and so on.

We now break the proof into two cases based on the values taken by the first 13 bits of  $C^1$  (recall  $C^1$  is the first non-zero column to the right of a band of zero columns).

1. **(Non-pathological Case):** Assume  $C^1$  is non-pathological, that is not all of its first 13 bits are zero. Then by a computer program it can easily be verified that the combined weight of Columns  $C^1$ ,  $C^2$  and  $C^3$  is at least 25.



2. **(Pathological Case):** Assume  $C^1$  is pathological i.e., each of its first 13 bits is zero. We now make the following easy claim.

**Claim C.1** *If  $C^1 = \langle x_i \rangle_{i=0}^{59}$  is pathological, then  $x_0 = x_1 = \dots = x_{56} = 0$ .*

*Proof:* Since  $x_0 = x_1 = \dots = x_{12} = 0$  (by definition), setting  $i = 13$  in Equation (13) yields  $x_{13} = 0$ . Similarly setting  $i = 14, \dots, 56$  gives  $x_{14} = x_{15} = \dots = x_{56} = 0$ . ■

Note that a pathological column does not contribute much to the weight of the codeword. Now denote the columns to the right of  $C^2$  by  $C^3, C^4$  and so on. Next consider  $C^2$ . Assume for the moment that it is pathological. Then by the same argument as in Claim C.1 (and Equation (14)), it holds that  $y_0 = y_1 \dots = y_{53} = 0$  (set  $i = 13, \dots, 56$  and note that  $x_i = 0$  for these values). In general, in a sequence of pathological columns (assume for the moment that this sequence has less than 12 columns) the  $i^{\text{th}}$  pathological column has the first  $60 - 3 \cdot i$  entries zero.

Assume  $C^{m+1}$  is the first non-pathological column (if any). So, if there are exactly  $m$  (for the moment assume  $m \leq 12$ ) pathological columns, then the column  $C^{m+1}$  (note that  $C^{m+1}$  cannot be all zero column by Equation (14)) must have a nonzero entry in the first  $60 - 3 \cdot (m + 1)$  entries. This is equivalent to it having a nonzero entry in the first 13 bits. Since otherwise an argument similar to Claim C.1 can be used to show that all the initial  $60 - 3(m + 1)$  bits are zero. The good thing is that a non-pathological column has a reasonably good weight. We now divide the remaining proof into two cases based on the number of consecutive pathological columns.

- (a) **(Number of consecutive pathological columns is at most 10):** In this case, we restrict ourselves to the case where there are 10 or less pathological columns. In this case, the combined weight of the pathological columns and at most two following non-pathological columns can be verified by a computer program to be at least 25.
- (b) **(Number of consecutive pathological columns is at least 11):** If there are a sequence of 11 or more pathological columns, then they already contribute more than 25 as verified by a computer search.

Hence 25 is the lower bound on the last 60 words of the SHA-1 message expansion code. ■

For completeness, we outline below the (combined) search pseudo-code for the Case 1 and Case 2(a).

