A Simple and Provably Good Code for SHA Message Expansion

Charanjit S. Jutla
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598
csjutla@watson.ibm.com

Anindya C. Patthak*
University of Texas at Austin
Austin, TX 78712
anindya@cs.utexas.edu

Abstract

We develop a new computer assisted technique for lower bounding the minimum distance of linear codes similar to those used in SHA-1 message expansion. Using this technique, we prove that a modified SHA-1 like code has minimum distance at least 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 propose a new compression function which is identical to SHA-1 except for the modified message expansion code. We argue that the high minimum weight of the message expansion code makes the new compression function resistant to recent differential attacks.

1 Introduction

Recall the SHA-1 message expansion code: 512 information bits are packed into 16 32-bit words $\langle W_0, \cdots, W_{15}\rangle$, and 64 additional words are generated by the recurrence:

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

The 80 words $\langle W_0, \cdots, W_{79}\rangle$ 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 64 words is only 30. This has been exploited in [WYY05b] to give a differential attack on SHA-1 with complexity 2^{69} hash operations.

In this paper, we show that it is possible to devise codes similar to the above code of SHA-1, but with a much better minimum distance. We give a computer assisted proof that the following code has minimum distance 82, and that too in just the last 64 words:

$$W_i = \begin{cases} W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15}) <<< 13) & \text{if } 16 \le i < 36 \\ W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15} \oplus W_{i-20}) <<< 13) & \text{if } 36 \le i \le 79 \end{cases} \tag{2}$$

Of course, since the dimension of this code is 32×16, a brute force search of $2^{32\times 16}$ is infeasible. Thus, we have to come up with an intelligent search, and prove that all $2^{32\times 16}$ cases have been considered. Not all such codes are amenable to such a tractable search, which in our case is about

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

2^{48} computer instructions. Thus, we have to carefully pick the coefficients of the above parity check equations, so as to keep the search feasible and the minimum distance large.

We next propose a new variant of SHA-1, which replaces the SHA-1 message expansion code with the above code. We argue below that this leads to a compression function which is resistant to recent differential attacks. We also argue in Section 4 that this expansion code is better than the expansion code of SHA-256, for which there is no known provable lower bound on the minimum distance.

A preliminary evaluation has shown that the new proposed compression function has at most a 5% overhead in speed over SHA-1 in a software implementation, and at most a 10% overhead in gate count in a high performance hardware implementation.

Recent attacks on MD-5 ([Riv92]), SHA-0 and SHA-1 (see [CJ98, BC04b, BC04a, WYY05a, WYY05b]) have capitalized on the poor message expansion of these compression functions. Essentially, all three hash functions follow the same underlying design principle: the 512-bit message is first expanded linearly 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.

The most effective attack against such compression functions is to launch a differential attack, where a difference in the messages leads to a zero difference in the output of the block cipher, thus leading to a collision. Unfortunately, in MD-5, SHA-0 and SHA-1, it is possible to start with a message difference which leads to a small difference in the N expanded keys. This in turn allows for a manageable overall differential characteristic of the above kind, hence leading to a collision attack.

In particular, in MD-5 a 3 bit difference in the 512-bit message leads to a difference of only 12 bits in the expanded ($N = 64$) keys. In SHA-0, there exists a message difference which leads to a 28 bit difference in the expanded ($N = 80$) keys. It turns out that the differential characteristic corresponding to the first 16 (and sometimes even first 20) steps can be assured with probability 1. Thus effectively, only the differences in latter steps contribute to lowering the probability of the differential characteristic holding. In SHA-0, the difference in the last 60 keys can be as low as 17 bits. Similarly, in SHA-1, there exists a message difference which leads to only a 27 bit difference in the last 60 keys.

Thus, the main reason that these hash functions have been undermined is their poor message expansion. With the new proposed code, any difference in messages leads to at least 82 bits of difference in the latter 64 keys. These (at least) 82 bit differences are injected into the update function of SHA-1 in the latter 64 steps, and any differential characteristic must account for canceling all (or most) of these differences. A useful heuristic that is often used in the analysis of SHA-0 and SHA-1 is that each bit difference in the key (in the latter 64 rounds) lowers the probability of success on average by a factor of $2^{2.5}$. Thus, we expect our proposed compression function to have a differential collision characteristic of probability close to $2^{-82\times 2.5}$. We also prove that the minimum weight of our proposed code in the last 60 keys is at least 75. The technique is general enough to obtain lower bounds on minimum weight of further front truncations. Note that, because of the change in the recurrence relation at $i = 36$, the codewords restricted to say the last 56 words, cannot be described as easily as the recurrence relation in Equation 2.

Organization: The rest of the paper is organized as follows: In section 2 we briefly review SHA-0, SHA-1. In section 3 we propose a new code and prove that it has good minimum distance.

We then use this new code to propose SHA1-**IME**, a modified version of SHA-1. In section 4 we compare SHA1-**IME** with SHA-256 ([Uni02]) and then make a few concluding remarks.

2 SHA-0 and SHA-1

2.1 SHA-0 Message Expansion Code

In this sub-section we describe the message expansion scheme used in SHA-0. Let $\langle M_0, \cdots, M_{15}\rangle$ be the 512 bits input to SHA, where each M_i is a word of 32 bits. Then the message expansion phase of SHA-0 outputs 80 words $\langle W_0, \cdots, W_{79}\rangle$ that are computed as follows:

SHA-0 :

$$W_i = M_i \quad \text{for } i = 0, 1, \cdots, 15, \text{ and}$$
$$W_i = W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \quad \text{for } i = 16, \cdots, 79. \tag{3}$$

Notice that the above can be seen as a linear code. Also notice that the expansion process applied to different bits is independent, that is there is no interleaving. This in fact makes the code rather weak and SHA-0 an easier target for the differential collision attack. Not surprisingly then that collision (and near-collision) attacks on SHA-0 have been the most successful in recent years (see [CJ98, BC04b, WYY05a]).

2.2 SHA-1 Message Expansion Code

Two years after the standard was set to SHA-0 [Uni93], an addendum was released in [Uni95], altering the message expansion scheme, and thus setting the standard to SHA-1. The change was attributed to correcting a technical weakness though no formal justification was given. The change may be interpreted as an attempt to improve the code by introducing mild interleaving. Precisely, the code in SHA-1 is the following: Let $\langle M_0, \cdots, M_{15}\rangle$ be the 512 bits input to SHA-1, where each M_i is a word of 32 bits. Then the message expansion phase outputs 80 words $\langle W_0, \cdots, W_{79}\rangle$ that are computed as follows:

SHA-1 :

$$W_i = M_i \quad \text{for } i = 0, 1, \cdots, 15, \text{ and}$$
$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) <<< 1 \quad \text{for } i = 16, \cdots, 79. \tag{4}$$

The notation "$<<< 1$" ("$<<< i$") denotes a one bit (i bit, respectively) rotation to the left. Note that the above code is linear too. Moreover if $\langle W_0, \cdots, W_{79}\rangle$ is a codeword, then so is $\langle W_0 <<< j, \cdots, W_{79} <<< j\rangle$ for all $j = 1, 2, \cdots, 31$. This can further be interpreted as follows: view the code-word as

$$\langle W_0^0, W_1^0, \cdots, W_{79}^0, W_0^1, \cdots, W_{79}^1, \cdots, W_{79}^{32}\rangle,$$

where W_i^j denotes the j^{th} bit of W_i. Then it is clear that this code is invariant under a rotation of 80 bits. These linear codes, a natural generalization of cyclic codes, are known as **quasi-cyclic**

codes in the literature. Quasi-cyclic codes have been studied extensively over the last 40 years. (See [TW67, Che92, Lal03, LS05] and the references therein.)

Unfortunately, the interleaving process in SHA-1 is not quite good. This is observed independently in [RO05] and in [MP05]. To explain it further we rewrite Equation 4 as follows:

$$\forall i, 0 \leq i \leq 63, \quad W_i = W_{i+2} \oplus W_{i+8} \oplus W_{i+13} \oplus (W_{i+16} >>> 1), \tag{5}$$

where "$>>> 1$" ("$>>> i$") denotes a one bit (i bit respectively) 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. This observation allows the authors in [BC04a, RO05, MP05] to generate low-weight differential patterns. These patterns are then used to create collisions or near-collisions in reduced version of SHA-1 with complexity better than the birthday-paradox bound. Extending this further [WYY05b] reports the first attack on the full 80-step SHA-1 with complexity close to 2^{69} hash functions. In there, the authors critically observe that the code not only has small weight codewords (≤ 44, [RO05, WYY05b]) but also that these small weight codewords are even sparser in the last 60 words (for example, [WYY05b] reports a codeword with weight 27 in the last 60 words).

3 SHA1-IME: A modified SHA proposal with a provably good code

In this section we propose a new hash function SHA1-**IME** (**IME** stands for "**I**mproved **M**essage **E**xpansion"). We use the same state update transformation as in SHA-1 or SHA-0. However, we replace the SHA-1 message expansion code by an equally simple code that has minimum distance provably at least 82, and that too in the last 64 words. The code, we denote it by $\mathcal{C}$, can be described as follows: Let $M_0, \cdots, M_{15}$ be the input message blocks. Then

SHA1-IME :
for $i = 0, 1, \cdots, 15, \quad W_i = M_i$ and
for $i = 16$ to 79

$$W_i = \begin{cases} W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15}) <<< 13) & \text{if } 16 \leq i < 36 \\ W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15} \oplus W_{i-20}) <<< 13) & \text{if } 36 \leq i \leq 79 \end{cases} \tag{6}$$

We now briefly describe the state update function used in SHA-1 (for details see [Uni95]). It comprises of total 80 steps divided in four rounds. Five 32-bits registers, conveniently denoted as A, B, C, D and E, are used. Their initial state is fixed and we denote it by $\langle A_0, B_0, C_0, D_0, E_0 \rangle$ (and in general, $\langle A_i, B_i, C_i, D_i, E_i \rangle$ after i steps). At step i, W_i is used to alter the state of these registers. Each step uses a fixed constant K_i and a bit-wise boolean function f_i that depends on the specific round. Formally, for $i = 0$ to 79,

$$\begin{aligned}
A_{i+1} &= W_i + (A_i <<< 5) + f_i(B_i, C_i, D_i) + E_i + K_i, \\
B_{i+1} &= A_i, \\
C_{i+1} &= B_i <<< 30, \\
D_{i+1} &= C_i, \\
E_{i+1} &= D_i,
\end{aligned}$$

Round	Step(i)	$f_i(X, Y, Z)$
1	0-19	$XY \vee \overline{X}Z$
2	20-39	$X \oplus Y \oplus Z$
3	40-59	$XY \oplus XZ \oplus YZ$
4	60-79	$X \oplus Y \oplus Z$

where '$+$' denotes the binary addition modulo 2^{32}.

We propose the following modified version of SHA-1 : SHA1-**IME**. In the message expansion phase it uses the code described in Equation 6. Then it uses the same state update function. How does SHA1-**IME** perform compared to existing SHA-1? It is virtually the same. We used a Pentium(R) 4, 3.06 GHz machine to execute 2^{28} many hash functions. The existing SHA-1 took

```
time in sec:  567.016000, time per sha1:2.112299e-06
```

whereas SHA1-**IME** took

```
time in sec: 585.719000, time per sha2: 2.181973e-06
```

We stress that the performance of the new hash operation remains virtually the same.

3.1 Intuition behind the code

As mentioned in subsection 2.2, Equation 5 shows that the SHA-1 code does not propagate well across different bit positions. One way to remedy this situation is to let $W_i = (W_{i+2} >>> 1) \oplus W_{i+8} \oplus W_{i+13} \oplus (W_{i+16} >>> 1)$. Now Equation 4 becomes $W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-16}) <<< 1 \oplus W_{i-14}$. 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 3.3.

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 16×32, and which is invariant under column rotations. Moreover, suppose that the code is of the form

$$W_i = \sum_{j=1}^{16} a_j W_{i-j} + \left(\left(\sum_{j=1}^{16} b_j W_{i-j} \right) <<< 1 \right),$$

where $a_1, \cdots, a_{16}, b_1, \cdots, b_{16}$ 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 $j' < 16$ is the largest j such that $b_{j'}$ is non-zero. First note that if a column, say C^i, is zero, then in the column to its right, say C^{i-1}, C^{i-1}_k (for $k = 0$ to $15 - j'$) can take any value (i.e., are free variables), and the rest of the column C^{i-1} can be all zero. Further, the propagation to columns C^{i-2}, C^{i-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.

3.2 A lower bound on the minimum distance

In this subsection, we give a computer assisted proof to conclude that the code proposed in Equation 6 has minimum distance at least 82 in just the last 64 words. First of all observe that $\mathcal{C}$ (described in Equation 6) too is a quasi-cyclic code. To see this observe that viewed appropriately a rotation by 80 bits leaves the code invariant. Establishing lower bound on the minimum distance of a quasi-cyclic code is a hard problem and has drawn considerable attention (see [Che92, Lal03]). Unfortunately, when the index (that is the minimum amount of rotation that leaves the code invariant) is as large as 80 (or even 64), the presently known bound seems computationally infeasible. In general, 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]). An interesting approach is taken in [RO05] where they restrict their search by keeping most columns zero. This allows them to find a codeword with low weight for SHA-1; however, they do not give a technique to lower bound the minimum weight of such codes.

Secondly, observe that the code $\mathcal{C}$ in SHA1-**IME** uses a left rotation by 13 bit. However, it is easy to see that as long as the amount of rotation is relatively prime to 32, the code remains the same up to a permutation of its columns. In particular, its minimum weight does not change if left rotate by 13 is replaced by a left rotate by 1. Therefore instead of $\mathcal{C}$, we consider the following code $\mathcal{C}'$ which is equivalent up to a permutation in the codeword positions : Let $M_0, \cdots, M_{15}$ be the message blocks. Then

$$
\begin{aligned}
\text{for } i &= 0, 1, \cdots, 15, \quad W_i = M_i \quad \text{and} \\
\text{for } i &= 16 \text{ to } 79
\end{aligned}
$$

$$
W_i = \begin{cases} W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15}) <<< 1) & \text{if } 16 \le i < 36 \\ W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-15} \oplus W_{i-20}) <<< 1) & \text{if } 36 \le i \le 79 \end{cases} \tag{7}
$$

In fact the following explicit permutation applied to the columns in $\mathcal{C}$ yields $\mathcal{C}'$:

$$
\pi : \{0, 1, \cdots, 31\} \to \{0, 1, \cdots, 31\} \text{ where } j \mapsto (5 \cdot j) \bmod 32
$$

since 5 is the inverse of 13 modulo 32.

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 $M_0, \cdots, M_{15}$ be the message blocks. Then

$$
\begin{aligned}
\text{for } i &= 0, 1, \cdots, 15, \quad W_i = M_i \quad \text{and} \\
\text{for } i &= 16 \text{ to } 63
\end{aligned}
$$

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

We first prove that this is indeed sufficient.

Lemma 3.1 *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 $U = \langle U_0, \cdots, U_{79} \rangle$. Denote $X = \langle U_0, \cdots, U_{15} \rangle$ and $Y = \langle U_{16} \cdots, U_{31} \rangle$ and $Z = \langle U_{32} \cdots, U_{79} \rangle$. Therefore $U = \langle X, Y, Z \rangle$. From Equation 7 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 $Y = \langle U_{16}, \cdots, U_{31} \rangle$, 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 $\mathcal{C}64$ 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 X. Therefore the minimum weight of $\mathcal{C}64$ is exactly the minimum weight of code $\mathcal{C}'$ in its last 64 words. Since $\mathcal{C}$ and $\mathcal{C}'$ is the same code up to a permutation of the co-ordinate positions, the minimum weight of $\mathcal{C}64$ is exactly the minimum weight of code $\mathcal{C}'$ in its last 64 words. (Observe that the permutation permutes only the columns, that is i^{th} word in $\mathcal{C}$ translates into the i^{th} permuted word of $\mathcal{C}'$.) ■

Next we prove a lower bound on the minimum distance of $\mathcal{C}64$. We break down the proof into several sub-cases. In each sub-case, we argue often following an exhaustive search over a small space that the minimum weight of the code is at least 82. We mention that a naive algorithm may require to search a space as large as $2^{32\times 16}$ which is clearly not feasible. Therefore the novelty in our approach lies in a careful sub-division of the problem into a small number of tractable cases. We mention that this approach is very general and may be used to give lower bounds on the minimum distance of similar quasi-cyclic codes or nearly-quasi-cyclic codes.

Theorem 3.2 *The code $\mathcal{C}64$ as defined by Equation 8 has minimum distance at least 82.*

Proof: It is easy to notice that the code $\mathcal{C}64$ is a quasi-cyclic code by noting that it is invariant under a 64 bit cyclic shift. From now onwards, we view the codewords of $\mathcal{C}64$ as a matrix that has 32 columns where each column is 64-bit long. The quasi-cyclic property then just mean 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. Since the code is a linear code, it suffices to prove that it has weight at least 82. 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 $C^1, C^2, \cdots, C^{31}$. Note that all C^i's are non-zero. In this case the following claim holds.

   Claim 3.3 *For any non-zero column C^i, there exists $k, 0 \leq k \leq 7$ such that the combined weight of columns $C^i, C^{i+1}, \cdots, C^{i+k}$ 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 B for an example). ■

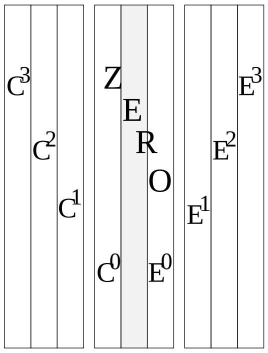
   Next we create a partition of the 32 columns into several groups. We pick a non-zero column C^i. Now following Claim 3.3, there exists $(k+1)$-columns $(0 \leq k \leq 7)$ such that the average weight of each column is at least 3. Consider the smallest k that achieves this. Then put these $(k+1)$ columns $C^i, C^{i+1}, \cdots, C^{i+k}$ into a group. Call these columns good columns and the group a good group. We then choose C^{k+i+1} 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 following Claim 3.3, e could be at most 7. Therefore the total weight of the codeword is at least

   $$3 \cdot (32 - e) + e = 96 - 2 \cdot e \geq 82.$$

2. (**At Least One Column Zero Case:**) Assume that there is at least one zero column. 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 $C^1, C^2, \cdots$ (see figure).

Also, going towards the right of C^0, denote the first non-zero column by E^1 and thereafter $E^2, E^3, \cdots$. 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 82.



It will be immaterial in our analysis below if there are some non-zero columns between C^0 and E^0. All we require in our analysis is that C^0 and E^0 are zero.

Next consider $C^1, C^2, \cdots$. 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 $C^1 = \langle y_0, \cdots, y_{63} \rangle$. Then

$$\forall i, 16 \leq i \leq 63, \quad 0 = y_i + y_{i-3} + y_{i-8} + y_{i-14} + y_{i-16}. \tag{9}$$

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 $C^2 = \langle x_0, \cdots, x_{63} \rangle$. Then

$$0 = \begin{cases} x_i + x_{i-3} + x_{i-8} + x_{i-14} + x_{i-16} + y_{i-1} + y_{i-2} + y_{i-15} & \text{for } 16 \leq i \leq 19 \\ x_i + x_{i-3} + x_{i-8} + x_{i-14} + x_{i-16} + y_{i-1} + y_{i-2} + y_{i-15} + y_{i-20} & \text{for } 20 \leq i \leq 63 \end{cases} \tag{10}$$

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

$$0 = \begin{cases} w_{i-1} + w_{i-2} + w_{i-15} & \text{for } 16 \leq i \leq 19 \\ w_{i-1} + w_{i-2} + w_{i-15} + w_{i-20} & \text{for } 20 \leq i \leq 63 \end{cases} \tag{11}$$

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 $E^2 = \langle z_0, \cdots, z_{63} \rangle$. Then

$$0 = \begin{cases} w_i + w_{i-3} + w_{i-8} + w_{i-14} + w_{i-16} + z_{i-1} + z_{i-2} + z_{i-15} & \text{for } 16 \leq i \leq 19 \\ w_i + w_{i-3} + w_{i-8} + w_{i-14} + w_{i-16} + z_{i-1} + z_{i-2} + z_{i-15} + z_{i-20} & \text{for } 20 \leq i \leq 63 \end{cases} \tag{12}$$

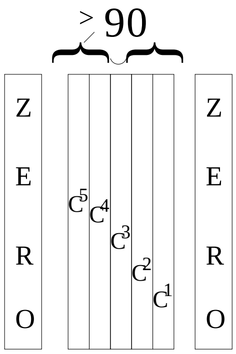
The following claim shows that at least four consecutive columns have to be non-zero.

Claim 3.4 *If C^0 is everywhere zero, and C^1 is non-zero, then so is C^2, C^3 and C^4.*

Proof: Suppose for a j it is the case that $C^j = E^1$, i.e., C^{j+1} is all zero. Then a homogeneous system of linear equations over $\mathbb{F}_2$ can be set up. Consider the $64 \times j$ variables in column C^1 through C^j. There are 48 equations for each of the columns C^1 through C^j. Also, there are 48 more equations for C^{j+1}. It is well known that such a system can have a non-trivial solution if and only if the rank of the co-efficient matrix is strictly smaller than the number of variables. It can easily be verified by a computer program that for $j = 1, 2, 3$, the system has full rank, that is exactly $64 \times j$. This can also be proved algebraically for $j = 1, 2$. We give a simple algebraic proof in the appendix (see Appendix A). ∎

This proof also highlights that for the rank to be full the recurrence relation must satisfy nice properties. Ranks of all linear systems considered in this paper have been computed using Gaussian elimination. We now divide the proof into two cases.

(a) (**Number Of Consecutive Non-Zero Columns Is At Most Five**):
By the claim above, we can safely assume that we have at least four consecutive non-zero columns. Also, if we assume $C^4 = E^1$, then the number of nontrivial solutions can be at most $2^{16} - 1$ (since the co-rank or nullity of the matrix is 16, as verified by implementing a Gaussian elimination program). Similarly, assuming $C^5 = E^1$, the number of nontrivial solutions can be at most $2^{32} - 1$. We do an exhaustive search to conclude that the minimum weight in the latter case is at least 90. (Note that this latter case alone is sufficient.)

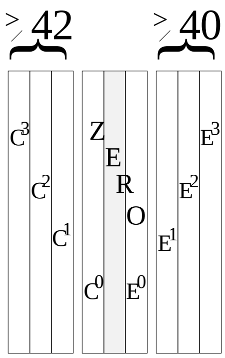


Case 2(a)

(b) (**Number Of Consecutive Non-Zero Columns Is At Least Six**): If case 1 and case 2(a) do not hold then, the only case that remains to be considered is the one where at least six consecutive columns are non-zero. Note that C^1, C^2, C^3 are then distinct from E^1, E^2, E^3. We use a computer program to verify that in this case the combined weight of C^1, C^2 and C^3 is at least 42.
Now recall Equation 11, the constraints induced on E^1. A quick observation reveals that its free variables are the first 15 bits and the very last bit. Depending on the values taken by E^1s first 15 bits we sub-divide our proof into two cases:

i. (**Non-Pathological Case:**) Here not all the first 15 bits of E^1 are zero.

This is the simpler case. In this case, the recurrence induces a good expansion. By an exhaustive search we obtain that in this case the combined weight of E^1, E^2 and E^3 is at least 40. Since the combined weight of C^1, C^2 and C^3 is at least 42, and that C^i, E^i are all distinct, together they establish this case.



Case 2(b)i

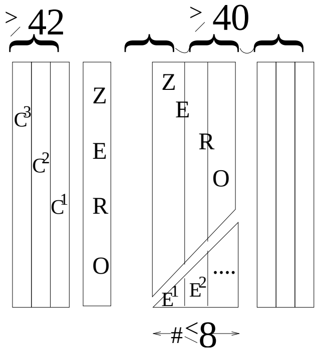
ii. (**Pathological Case:**) Here we assume that the first 15 variables of E^1 are all zero. This is the most subtle and difficult case. Going back to Equation 11, we note that in this case it must hold that $w_{63} = 1$ and for all $0 \le i \le 62, w_i = 0$. We call such w *pathological*.
Now consider Equation 12. 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 12 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., $E^3, E^4, \cdots$ 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^{st} bit, that of E^4 will be 60^{th} bit and so on. Also each column will have a free variable in its 63^{rd} 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.
We now sub-divide into many small cases depending primarily on the number of pathological columns (and thus on the number of free variables).

A. (# **Pathological Columns** ≤ 8) We break this case into two sub-cases. That each of these sub-cases holds has been verified using a computer program.

(I). 6^{th} **and earlier non-pathological columns are non-zero** :

In this case, we verify that the combined weight of the pathological columns and the first three non-pathological columns to the right of the pathological columns is at least 40. This ensures that in this case the minimum weight is at least 82.



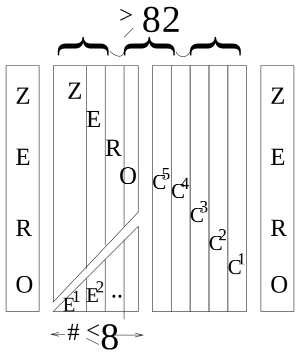
Case 2(b)(ii)(A)(I)

We mention that the search space dimension can be estimated as

$$\# \text{ of Pathological variables} + \# \text{ of Non-Pathological Columns} \times 16,$$

which is at most 40 in this case.
We next consider the case where the non-pathological columns are same as one of C^1, C^2 or C^3.

(II). $\mathbf{6^{th}}$ **or earlier non-pathological column is identically zero**: Firstly note that it suffices to check the case where the 6^{th} non-pathological column is identically zero (that is $E^3 = C^3$), since other cases do fall in this case.
Now we consider the parity check equations induced on the pathological columns and the six non-pathological columns. Note that C^1 satisfies Equation 9 and that E^1 satisfies Equation 11. Also note that in between columns satisfy equations similar to Equations 10 and 12. These equations then set up a homogeneous system of linear equations whose nullity can be verified (by a computer program) to be at most 40.



Case 2(b)(ii)(A)(II)

Let the number of pathological columns be p and the number of non-pathological columns be n. Specifically then the nullity of the system can then be shown to be exactly (see Appendix A Claim A.3)

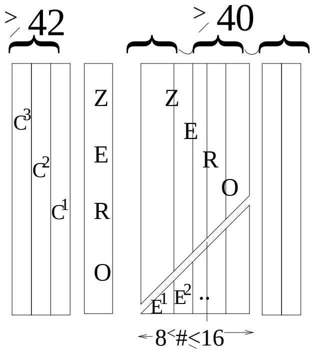
$$p + 64 \times n - 48 \times (n + 1) = p + 16 \cdot n - 48,$$

which is at most 40 in this case. We do an exhaustive search over the null space to establish that the min-weight is at least 82.

B. ($8 < \#$ **Pathological Columns** ≤ 16) We also break this case into two sub-cases. That each of these sub-cases holds has been verified using a computer program.

(I). 5^{th} **and earlier non-pathological columns are non-zero**

In this case, we verify that the combined weight of the pathological columns and the first two non-pathological columns to the right of the pathological columns is at least 40. This ensures that in this case the minimum weight is at least 82.

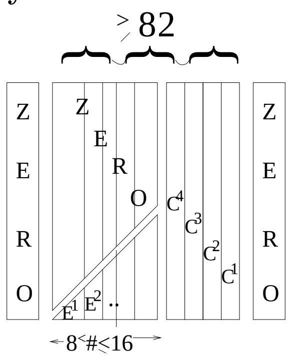


Case 2(b)(ii)(B)(I)

Therefore the case that remains to be considered is the one where the non-pathological columns are same as one of C^2 or C^3 which leads us to the next case.

(II). **5^{th} or earlier non-pathological column is identically zero**:

Firstly, note that it suffices to check the case when the 5^{th} non-pathological column is identically zero (that is $E^2 = C^3$), since other cases do fall in this case. As in the 2^{nd} sub-case of the previous case (i.e., Case 2(b)(ii)(A)(II)), we verify that the min-weight is at least 82.

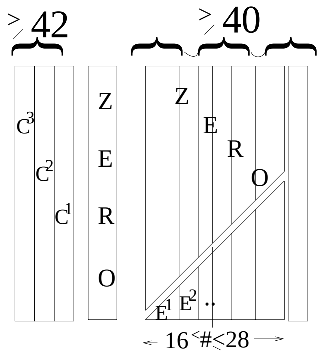


Case 2(b)(ii)(B)(II)

C. ($16 <$ **Pathological Columns** ≤ 28) First of all, notice that 28 columns is enough, since by our assumption there is at least one zero column and three non-pathological column (i.e., C^1, C^2, C^3). Now, we also break this case into two sub-cases. That each of these sub-cases holds has been verified using a computer program.

(I). 4^{th} **and earlier non-pathological columns are non-zero**

In this case, we verify that the combined weight of the pathological columns and the first non-pathological column to the right of the pathological columns is at least 40. This ensures that in this case the minimum weight is at least 82.

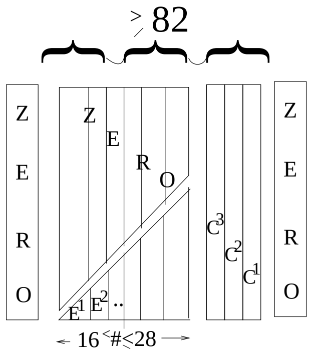


Case 2(b)(ii)(C)(I)

Therefore the case that remains to be considered is the one where the 1^{st} non-pathological column is the same as C^3.

(II). **4^{th} non-pathological column is identically zero**:

As in the 2^{nd} sub-case of the previous case (or Case 2(b)(ii)(A)(II)), we verify that the min-weight is at least 82.



Case 2(b)(ii)(C)(II)

We remark that the minimum weight of this code can at most be 82 and therefore our result is tight. We found the following codeword while searching for Case 2(b)(ii)(A)(II). Below we only give eight columns that includes six non-zero and two zero columns. The rests are all zero columns. Below the columns are placed horizontally.

```
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0011110010011110 1000000001101001 1101001001010110 0000110010010000
1011000101000100 0010111101001000 1011100010101100 1101000000101111
1010101000111011 0010100100110010 1000000101001000 0110011000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000100
0000000000000000 0000000000000000 0000000000000000 0000000000000011
0000000000000000 0000000000000000 0000000000000000 0000000000000001
0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

3.3 The Last Sixty Words

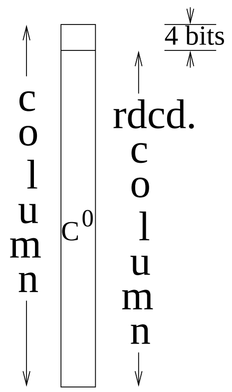
In this subsection, we prove that the minimum weight of the code $\mathcal{C}$ in the last 60 words is at least 75. In general, our proof strategy is robust, i.e., it can in principle be adapted to estimate the minimum weight of this code in the last $4 \cdot n$ (where n is an integer) number of steps, 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 3.3. 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 3.2, 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 3.5 *The code $\mathcal{C}64$, as defined by Equation 8, has minimum weight at least 75 in its last 60 words.*

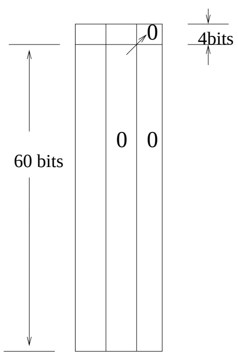
Proof: Consider any column of length 64 bits. A column restricted to its bottom most 60 bits will henceforth be referred to as a *reduced column* (see figure).

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.



A Reduced Column

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 $C^1, C^2, \cdots, C^{31}$. 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 $x = \langle x_0, x_1, \cdots, x_{63}\rangle$ and C^1 by $y = \langle y_0, y_1, \cdots, y_{63}\rangle$. Note that by the assumption $x_i = y_i = 0$ for all $i = 4, \cdots, 63$. Now consider the parity check equations of $\mathcal{C}64$ 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 3.6 *For any non-zero column C^i, there exists $k, 0 \leq k \leq 7$ such that the combined weight of the reduced columns $C^i, C^{i+1}, \cdots, C^{i+k}$ is at least $3 \cdot (k+1)$.*

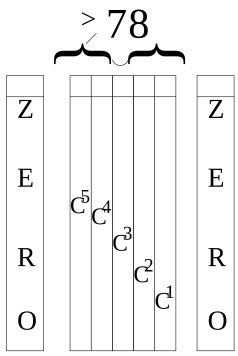
Note that although we restrict ourselves to at most 2 bits ON in reduced C^0, we must consider all 16 possibilities for the first 4 bits of C^0 to be able to define reduced column C^1 (from 16 bits in reduced column in C^1 and all the bits in C^0). Despite this the search is easily conducted.

Then, following the same line of argument as in Case 1 (**All Columns Non-Zero Case**) of Theorem 3.2, 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 3.2. 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.

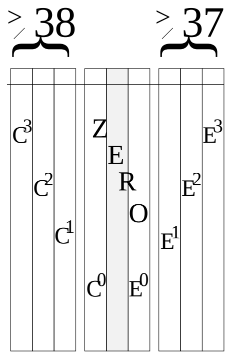


Case 3(a)

   (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.

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.



Case 3(b)(i)

ii. (**Pathological Case**)

A. **# of Pathological columns** ≤ 8

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

B. $8 <$ **# of Pathological columns** ≤ 16

(I). **5**th **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.

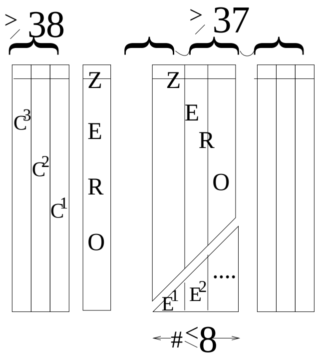
(II). **5**th **or earlier non-pathological column is zero**: The combined minimum weight of these reduced columns is at least 75.

C. $16 <$ **# of Pathological columns** ≤ 28

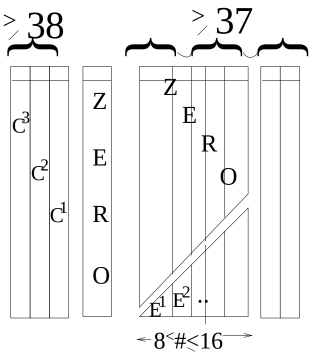
(I). **4**th **and earlier non-pathological columns are non-zero** : The combined weight of the pathological reduced columns and the first non-pathological reduced columns to the right of the pathological columns is at least 37.

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

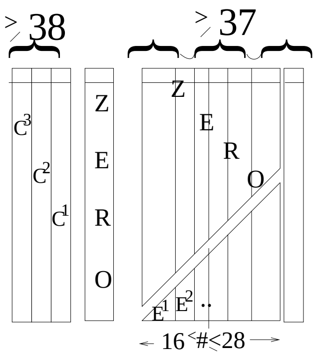
Therefore, in all these cases the combined weight of the reduced column is at least 75. This establishes the theorem. ■



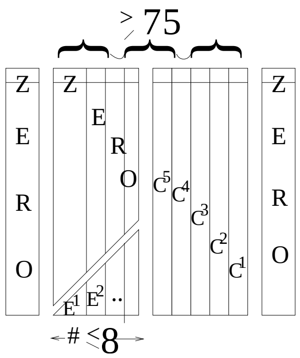
Case 2(b)(ii)(A)(I)



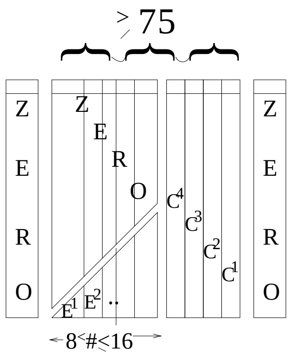
Case 2(b)(ii)(B)(I)



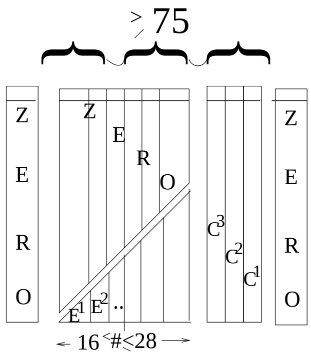
Case 2(b)(ii)(C)(I)



Case 2(b)(ii)(A)(II)



Case 2(b)(ii)(B)(II)



Case 2(b)(ii)(C)(II)

Various Cases in the proof of Theorem 3.5
(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.

3.4 The Last Forty-Eight Words

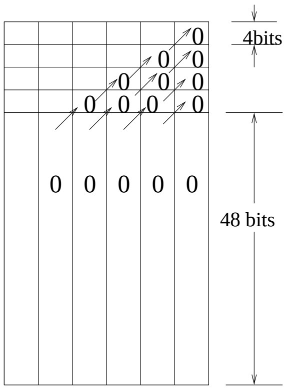
In this subsection, we prove that the code $\mathcal{C}64$ has minimum weight at least 52 in its last 48 words. As mentioned previously, this proof is more computation intensive as the dimension of the search space increases by an additive factor of 16. The good thing is that we need to show an average 2 per column, viz a viz Claim 3.3. 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 $\mathcal{C}64$ 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 3.5. Recall that in that proof (that is the proof of Theorem 3.5) 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 3.5, we consider reduced columns (here restricted to last 48 entries). We then can verify that under the assumption that all columns are non-zero, the reduced columns cannot be too sparse. This then allows us to prove the following theorem.

Theorem 3.7 *The code $\mathcal{C}64$ as defined by Equation 8 has minimum weight at least 52 in its last 48 words.*

Proof: Consider any column of length 64 bits. Here a column restricted to its bottom most 48 bits will henceforth be referred as a *reduced column.*
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 two main cases, depending on the existence of a zero column.

1. (**All Columns Are Non-Zero But Reduced Column Can Be Zero Case**): Consider any such codeword. Also consider any non-zero reduced column, w.l.o.g., let it be C^0. Denote the reduced columns, to the left of C^0 by $C^1, C^2, \cdots, C^{31}$. Note that if five consecutive reduced columns are zero, then the first column must be everywhere zero.

   This is easily obtained by setting i suitably in the parity check equations of the code $\mathcal{C}64$ (see figure). We handle that case latter. Therefore we can safely assume that no five consecutive reduced columns are zero.

   Then the following is easily verified by a computer program.

   Claim 3.8 *For any non-zero column C^i, there exists $k, 0 \leq k \leq 6$ such that the combined weight of the reduced columns $C^i, C^{i+1}, \cdots, C^{i+k}$ is at least $(k+1)$. Furthermore, there exists $\ell, 0 \leq \ell \leq 8$ such that the combined weight of the reduced columns $C^i, C^{i+1}, \cdots, C^{i+\ell}$ is at least $2 \cdot (\ell+1)$.*

   Note that although we restrict ourselves to at most 1 bit ON in reduced C^0, we must consider all 2^{16} possibilities for the first 16 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). Since we rely heavily on early stopping, these bits must be guessed in a lazy fashion to make the search feasible. Then following the same line of argument as in Case 1 (**All Columns Non-Zero Case**) of Theorem 3.5, it can be shown that the total weight of the reduced columns is at least 53 (since 24 columns yield at least 48 and the remaining eight columns yield at least 8, or 25 columns yield at least 50 and the remaining 7 yields 7, or 26 columns yield 52 and remaining 6 at least 1).

2. **At Least One Column Zero Case**: In this case the first column must be everywhere zero. This case can then be handled as the **Zero Case** in the proof of theorem 3.2. We consider the same number of cases and we count only the last 48 bits in a column. We remark that in each such cases, it can be shown that the weight in the last 48 rounds is at least 52. We skip the details.

4 Conclusion

4.1 Alternate codes

Notice that the code $\mathcal{C}64$ has a sliding window of size 20, that is to encode a message using this code, an LFSR would require 20 registers. The following code has a sliding of size 17. This may be useful for direct LFSR-type hardware implementation, since this would require three less registers than what the code $\mathcal{C}64$ requires.

Remark 4.1 *We mention here that using our technique, it can be shown that the following code has similar good minimum weight parameters as that of* $\mathcal{C}64$ *.*

Alternative1 *:*
for $i = 0, 1, \cdots, 15, \quad W_i = M_i$ *and*
for $i = 16$ *to 63*

$$W_i = \begin{cases} W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-11}) <<< 13) & \textit{if } 16 \le i < 17 \\ W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-11} \oplus W_{i-17}) <<< 13) & \textit{if } 17 \le i \le 63 \end{cases} \tag{13}$$

We expect the following code too to have equally good properties as the codes we have considered/mentioned previously. However, because of additional pathological variables, the analysis becomes more complex and we defer the complete analysis to a later time.

Remark 4.2 $\langle W_0, \cdots, W_{79} \rangle$ *are computed from the message* $\langle M_0, \cdots, M_{15} \rangle$ *as follows:*

Alternative2 *:*
for $i = 0, 1, \cdots, 15, \quad W_i = M_i$ *and*
for $i = 16$ *to 63*

$$\begin{aligned} W_i \ = \ & W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-11} \oplus W_{i-15}) <<< 1) \\ & \oplus ((W_{i-1} \oplus W_{i-2} \oplus W_{i-11} \oplus W_{i-15}) >>> 1) \end{aligned} \tag{14}$$

4.2 Our proposed code vs. SHA-256 code

The code in SHA-256 ([Uni02]) is the following: Let $\langle W_0, \cdots, W_{15} \rangle$ be the 512 bits input to SHA-256, where each W_i is a word of 32 bits. Then the message expansion phase outputs $\langle W_0, \cdots, W_{63} \rangle$ where

$$\forall i, 16 \le i \le 63, \quad W_i \ = \ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16}, \tag{15}$$

where σ_0 and σ_1 are as follows:

$$\sigma_0(x) \stackrel{def}{=} (x >>> 7) \oplus (x >>> 18) \oplus (x >> 3).$$

$$\sigma_1(x) \stackrel{def}{=} (x >>> 17) \oplus (x >>> 19) \oplus (x >> 10);$$

In the above, "$>> i$" denotes a right shift by i bit and '+' denotes binary addition modulo 2^{32}. Note that the binary addition makes the code non-linear. We do not see how to lower bound the minimum weight of the above code. In spite of the complex description, we do not know how to formally argue about the security that this code offers.

One property that the SHA-256 code has which might be useful against [CJ98] and [WYY05b] attacks is that the code is not quasi-cyclic. These attacks require that a codeword rotated (along columns) is again a codeword. Similarly, the attacks require that the codewords shifted (along rows) is again a codeword. In fact, even our proposed code, although quasi-cyclic, is not invariant under shifts along rows. This is because the recurrence relation changes from step 36 onwards. However, claiming security on this basis maybe short-lived, and arguably there is no substitute to actually proving that the code has a high minimum weight.

4.3 Modifying SHA-256

It should be noted that SHA-256, unlike SHA-1, has only 64 steps. There are two reasons why the designers of SHA-256 probably considered it safe to reduce the number of steps: firstly, since SHA-256 produces a 128 bit output, its non-linear block cipher has eight 32 bit registers instead of the five that SHA-1 has. This in turn means that any disturbance introduced using the expanded message words W_i carries on for at least eight rounds (instead of five), and hence the probability of forcing local collisions goes down. Secondly, the SHA-256 message expansion code itself is more involved and possibly has better minimum distance (though as discussed in the previous subsection, there is no proof of that).

Utilizing the first observation, we believe that a provably good message expansion into 64 words does indeed render the "modified" SHA-256 secure against differential attacks. For the code we can use a back truncation of the code $\mathcal{C}$ analyzed in this paper, i.e. given by equation (2) but with $i <= 63$. Of course, one would need to analyze this code from scratch, as the minimum weight numbers for the code $\mathcal{C}$ do not automatically yield numbers for the back truncation.

Another interesting code, which we plan to analyze in the future, is a code similar to Alternative 2 above but with a sliding window of size 20. Recall (see the last para of section 3.1) that increasing the window size allows us to get rid of certain pathological variables, and makes the search feasible. Moreover, it also simplifies the analysis considerably. In particular, the code we plan to analyze and recommend for SHA-256 is:

$$W_i = \begin{cases} W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} & \\ \oplus((W_{i-1} \oplus W_{i-2} \oplus W_{i-15}) <<< 13) & \\ \oplus((W_{i-1} \oplus W_{i-2} \oplus W_{i-15}) >>> 13) & \text{if } 16 \leq i < 36 \\ & \\ W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} & \\ \oplus((W_{i-1} \oplus W_{i-2} \oplus W_{i-15}) <<< 13) & \\ \oplus((W_{i-1} \oplus W_{i-2} \oplus W_{i-15} \oplus W_{i-20}) >>> 13) & \text{if } 36 \leq i \leq 63 \end{cases} \tag{16}$$

As before, we would need to lower bound its minimum weight in the last 48 (and possibly last

32) words. One crucial observation we make is that in analyzing $\mathcal{C}$, we could estimate the dimension of the subspace such that $C^5 = E^1$ (see case 2(a) in the proof of Theorem 3.2) to be about 32 (in fact exactly 32). This follows by just observing the number of variables, and the homogeneous equations involved. However, when the length of the code is reduced as above, this subspace has dimension at least 48. But, by mixing three columns at a time as in the above code, the number of equations in that case (i.e., in case 2(a) in the proof of Theorem 3.2) goes up considerably, and the null space has a more reasonable dimension of about 16.

4.4 Acknowledgment

We thank Pankaj Rohatgi for motivating us throughout this work, and particularly for his insightful comments on an earlier version of this manuscript. We also thank W. Eric Hall for providing us with the hardware estimates, and Ronald Rivest for suggesting the name SHA1-**IME**. The second author also wishes to thank Alexandar Vardy and Patrick Solé for helpful suggestions.

References

[BC04a] E. Biham and R. Chen. Near collisions of SHA-0. In *Crypto*, 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*, 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.

[Lal03] K. Lally. Quasicyclic codes of index ℓ over $\mathbb{F}_q$ Viewed as $\mathbb{F}_q[x]$-submodules of $\mathbb{F}_{q^l}[x]/\langle x^m - 1\rangle$. In *Lecture Notes in Computer Science, Vol. 2643, Springer*, 2003.

[LS05] Ling and P. Solé. Structure of quasi-clcyic 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.

[Uni93] United States Department of Commerce, National Institute of Standards and Technology, Federal Information Processing Standard Publication #180. *Secure Hash Standard*, 1993.

[Uni95] United States Department of Commerce, National Institute of Standards and Technology, Federal Information Processing Standard Publication #180-1 (addendum to [Uni93]). *Secure Hash Standard*, 1995.

[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.

[WYY05a] X. Wang, H. Yu, and Y. L. Yin. Efficient collision search attacks in SHA-0. In *Crypto*, 2005.

[WYY05b] X. Wang, H. Yu, and Y. L. Yin. Finding collisions in the full SHA-1. In *Crypto*, 2005.

A Rank proofs

Claim A.1 *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 following 48×64 dimensional parity check matrices (essentially Equations 9 and 11) over $\mathbb{F}_2$

$$\underbrace{\begin{pmatrix} 10100000010000100100000 & \cdots & 0000000000000000000 \\ 01010000001000010010000 & \cdots & 0000000000000000000 \\ \ddots & \cdots & \ddots \\ 00000000000000000000000 & \cdots & 0101000000100001001 \\ \end{pmatrix}}_{H_1}$$

$$\underbrace{\begin{pmatrix} 0100000000000011000000 & \cdots & 000000000000000000000 \\ 0010000000000001100000 & \cdots & 000000000000000000000 \\ 0001000000000000110000 & \cdots & 000000000000000000000 \\ 0000100000000000011000 & \cdots & 000000000000000000000 \\ \\ 1000010000000000001100 & \cdots & 000000000000000000000 \\ 0100001000000000000110 & \cdots & 000000000000000000000 \\ \ddots & \cdots & \ddots \\ 0000000000000000000000 & \cdots & 100001000000000000110 \\ \end{pmatrix}}_{H_2}$$

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^{th} through 20^{th} 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 $c = \langle c_0, \cdots, c_{63}\rangle$ in $\mathbb{F}_2^{64}$ the polynomial $c(s) = \sum_{i=0}^{63} c_i s^i$ in $\mathbb{F}_2[s]$. Then the following polynomials can be associated with the 1^{st} and 5^{th} rows of matrix H_1 and H_2, respectively:

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

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

Further note that the i^{th} (note $1 \leq i \leq 48$) row of H_1 then gets associated with $s^{i-1}p(s)$. Similarly the j^{th} (note we restrict ourselves to $5 \leq j \leq 20$) row of H_2 then gets associated with $s^{j-5}r(s)$. 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:

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

with additional constraints that degree(α) ≤ 47 and degree(β) ≤ 15. However, it is well known that $p(s)$ is irreducible, therefore if such a equation holds then it must be the case that $p(s)$ divides $r(s)$. However, it is easy to check that $p(s)$ does not divide $r(s)$, thus leading to a contradiction. Therefore H has full rank. ∎

Claim A.2 *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 degree(a) ≤ 3 and degree(b) ≤ 43. Since degree($q \cdot a$) $<$ 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

$$\left.\begin{array}{rcll} \alpha(x)p(x) + \beta(x)r(x) & = & 0 & (17) \\ x^4\beta(x)p(x) + \gamma(x)r(x) & = & 0 & (18) \end{array}\right| \begin{array}{l} \text{with restrictions} \\ \text{degree}(\alpha) \leq 47, \\ \text{degree}(\beta) \leq 43, \text{ and} \\ \text{degree}(\gamma) \leq 35. \end{array}$$

Since $p(x)$ is an irreducible polynomial, and $p(x) \nmid r(x)$, observe from Equation 17 that $p(x)|\beta(x)$. Hence, set $\beta(x) = \mu(x)p(x)$. Substituting in Equation 18 we get

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

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

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.3 *Let $E^1, E^2, \cdots, E^p$ be p pathological columns. Also, let $E^{p+1}, E^{p+2}, \cdots, E^{p+n}$ 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*

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

Proof: Let $N_{i,j}, (1 \leq i \leq n, 0 \leq j \leq 63)$ denote the entries in the non-pathological columns. Also let $P_{i,j}, (1 \leq i \leq p,$ for each i, $64 - i \geq j \leq 63)$ be the pathological variables. We will denote $N_i = \langle N_{i,0}, \cdots, N_{i,63}\rangle$ and $P_i = \langle P_{i,64-i}, \cdots, P_{i,63}\rangle$. Let $H_{1|i}$ denote the matrix H_1 restricted to the last i columns. (Note that only the last i rows will be non-zero.) Also let $H_{2|i}$ denote the matrix H_2 restricted to the last i columns. (Note that only the last $i-1$ rows will be non-zero.) Note that $\langle P_1, \cdots, P_p, N_1, \cdots, N_n\rangle$ must belong to the null space of the following matrix:

$$\mathcal{H} = \begin{pmatrix} H_{1|1} & H_{2|2} & & & & & & & \\ & H_{1|2} & H_{2|3} & & & & & & \\ & & \ddots & \ddots & & & & & \\ & & & H_{1|p-1} & H_{2|p} & & & & \\ & & & & H_{1|p} & H_2 & & & \\ & & & & & H_1 & H_2 & & \\ & & & & & & \ddots & \ddots & \\ & & & & & & & H_1 & H_2 \\ & & & & & & & & H_1 \end{pmatrix}$$

Note that when we restrict H_1 or H_2 to the last few columns, the top rows in that restricted entries may become zero row. We remove such rows if the entire row in the above matrix $\mathcal{H}$ becomes everywhere zero. Note that with this modification, the following sub-matrix is already in the echelon form:

$$\mathcal{H}1 = \left.\begin{pmatrix} H_{1|1} & H_{2|2} & & & \\ & H_{1|2} & H_{2|3} & & \\ & & \ddots & \ddots & \\ & & & H_{1|p-1} & H_{2|p} \end{pmatrix}\right\} (p-1) \text{ blocks}$$

(Observe that first block corresponding to $(H_{1|1}\ H_{2|2})$ reduces to (1 10), and that corresponding to $(H_{1|2}\ H_{2|3})$ reduces to $\begin{pmatrix} 10 & 100 \\ 01 & 110 \end{pmatrix}$.)

Furthermore, since by assumption the following sub-matrix has full rank:

$$\mathcal{H}2 \; = \; \left.\begin{pmatrix} H_2 & & & \\ H_1 & H_2 & & \\ \ddots & \ddots & & \\ & & H_1 & H_2 \\ & & & H_1 \end{pmatrix}\right\} (n+1) \text{ blocks}$$

the matrix $\mathcal{H}$ has full rank. Note here that in the top $48 - p$ rows, $H_{1|p}$ is entirely zero. However these rows in $\mathcal{H}$ are independent since $\mathcal{H}2$ has full rank. In the remaining rows $H_{1|p}$ is in echelon form and hence independent. Note that it has number of rows i.e., constraints:

$$48 \times (n+1) + \sum_{i=1}^{p-1} i = 48(n+1) + \frac{p(p-1)}{2}.$$

Also, note the number of variables i.e., columns is

$$64 \times n + \sum_{i=1}^{p} i = 64 \cdot n + \frac{p(p+1)}{2}.$$

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.

$\blacksquare$

B Examples

We cite below an example where over 7 columns an average of 3 does not hold. Below we only give 8 columns and the columns are placed horizontally. Note that the 8 columns yield 29, whereas the first 7 columns yield only 14.

```
0000000000000000000000000000000000000000000000000000000001000000
0000000000000000000000000000000000000000000000000000000000110110
0000000000000000000000000000000000000000000000000000000000010100
0000000000000000000000000000000000000000000000000000000000001110
0000000000000000000000000000000000000000000000000000000000000100
0000000000000000000000000000000000000000000000000000000000000011
0000000000000000000000000000000000000000000000000000000000000001
1000101010000000001001000010000010000100101100000010001000010000
```