A Matching Lower Bound on the Minimum Weight of SHA-1 Expansion Code

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

Recently, Wang, Yin, and Yu ([WYY05b]) have used a low weight codeword in the SHA-1 message expansion to show a better than brute force method to find collisions in SHA-1. The smallest minimum weight codeword they report has a (bit) weight of 25 in the last 60 of the 80 expanded words. In this paper we show, using a computer assisted method, that this is indeed the smallest weight codeword. In particular, we show that the minimum weight over $\mathbb{F}_2$ of any non-zero codeword in the SHA-1 (linear) message expansion code, projected on the last 60 words, is at least 25.

1 Introduction

Recently, a sequence of results ([CJ98, WFLY04, BC04a, BC04b, WYY05a]) culminating in the celebrated work of [WYY05b] has shown a method to find collisions in SHA-1 with only 2^{69} (SHA-1) hash operations. This is better than the 2^{80} hash operations required to find a collision using the birthday attack. One key ingredient of their result is a low weight codeword in the SHA-1 (linear) message expansion code, which they found using a computer search. This codeword (of length 80 32-bit words) has a (bit) weight of only 25, when counting only the bits ON in the last 60 words. In [RO05, MP05] the authors using computer assisted methods, report similar low weight codewords. However, they give no lower bound.

A useful heuristic that is often used (and we stress that the actual analysis is much more complex) is that each bit which is ON in the last 60 words contributes to lowering the success probability of the attack by $2^{2.5}$. Using the 25 weight codeword, one can then estimate the probability of success of the attack to be about 2^{-62} (they actually use a 27 weight codeword, as some other conditions need to be met as well).

The question then naturally arises as to whether there are other low weight codewords lurking, especially of weight less than 25 in the last 60 words. In this paper we settle this open problem in the negative. In a recent paper [JP05], we have developed a novel computer assisted technique to lower bound the minimum weight of SHA-1 like message expansion codes. The code considered in

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

that paper yields a minimum weight of 72 in the last 60 words, which we argue makes a modified SHA-1 (called SHA1-**IME**) immune to recent differential attacks.

The SHA-1 code by comparison is relatively simple, and since we need to prove a lower bound of only 25, the method becomes much simpler. However, this serves as a nice example (and a primer) of how the general technique works for proving much better lower bounds.

2 SHA-1 Message Expansion Code and A Lower Bound

We begin by recalling the message expansion code in SHA-1 ([Uni95]). 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 $\langle W_0, \cdots, W_{79}\rangle$ which are computed as follows:

SHA-1 :
$W_i = M_i$ for $i = 0, 1, \cdots, 15$, and

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

The notation "<<< 1" ("<<< i") denotes a one bit (i bit, respectively) rotation to the left. Note that the above is a linear code.

Unfortunately, the above message expansion code in SHA-1 is not quite satisfactory. This is observed independently in [RO05] and in [MP05]. To explain it further we rewrite Equation 1 as follows:

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

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. Particularly in [WYY05b] the authors report a codeword in SHA-1 message expansion code that has weight 25 in the last 60 words.

In there, it was left open whether a lower weight codeword (in the last 60 words) exists in the code. We next prove (with computer assistance) a matching lower bound that 25 is indeed optimal, i.e., no codeword with weight less than 25 in the last 60 words exists in the SHA-1 message expansion code.

Theorem 2.1 *SHA-1 message expansion code has minimum weight 25 in the last 60 words.*

Proof: We employ the proof technique introduced in [JP05]. First observe that it suffices to consider the code of length 60 given by the recurrence relation

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

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 :

$$\text{for } i = 13 \text{ to } 56 \quad x_i \oplus x_{i-5} \oplus x_{i-11} \oplus x_{i-13} = 0. \tag{3}$$

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,

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

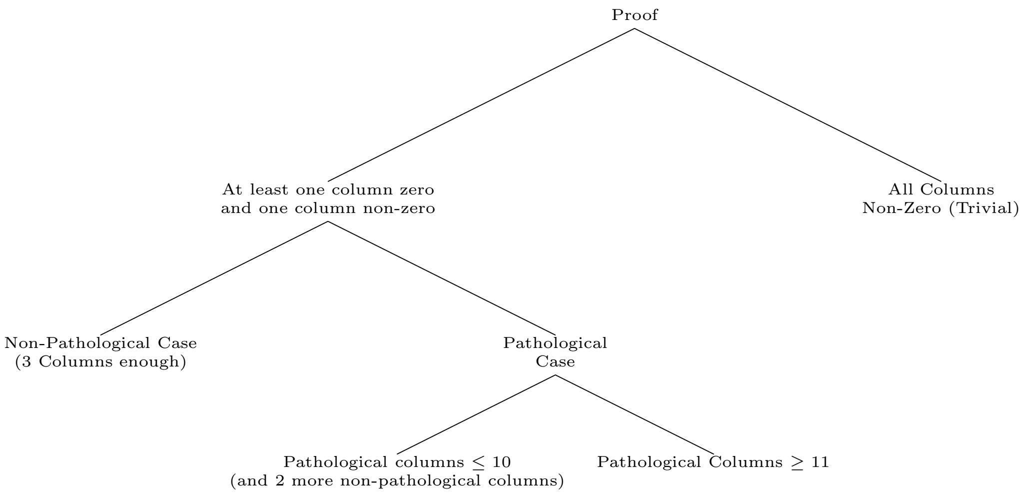
which can be rewritten as

$$\text{for } i = 13 \text{ to } 56 \quad y_i \oplus y_{i-5} \oplus y_{i-11} \oplus y_{i-13} = x_{i+3}. \tag{4}$$

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

   Proof: Since $x_0 = x_1 = \cdots = x_{12} = 0$ (by definition), setting $i = 13$ in Equation 3 yields $x_{13} = 0$. Similarly setting $i = 14, \cdots, 56$ gives $x_{14} = x_{15} = \cdots = x_{56} = 0$. ■

   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 2.2 (and Equation 4), it holds that $y_0 = y_1 \cdots = y_{53} = 0$ (set $i = 13, \cdots, 56$ 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^{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 4) 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 2.2 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).

1. Choose the number m of pathological columns ($0 \leq m \leq 10$). For each pathological column choose the last three bits of that column. The other bits are determined by these bits recalling that in the i^{th} column, the first $60 - 3 \cdot i$ bits are zero.
2. Now choose the first 13 bits of the first non-pathological column (and also choose its last three bits). From these bits all its remaining bits can be determined. If the total count is ≥ 25, then go to the next choice in Step (1); otherwise do Step (3).
3. Choose the first 13 bits of the next column (and its last three bits), from which all its other bits can be determined. If the count is ≥ 25, then go to the next choice in Step (1); otherwise do Step (4).
4. Choose the first 13 bits of the next column (and its last three bits), from which all its other bits can be determined. If the count is < 25, output **FAIL**; otherwise goto the next choice in Step (1).

While running the above, we found three codewords with weight 25, which are all listed below. The first one is reported earlier in [WYY05b]. The columns are listed horizontally, and the leftmost column is the top most column. Note that each of them has five pathological columns. The pathological columns are separated from the non-pathological columns by a blank line.

```
Codeword1
 pathological count= 6,cnt1 =15, cnt2= 4, cnt3== 0, sum =25::
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000001
000000000000000000000000000000000000000000000000000000001000
000000000000000000000000000000000000000000000000000001000000
000000000000000000000000000000000000000000000000001000010000
000000000000000000000000000000000000000000000001000000000000

010101100110001101100100010101010000000000000000000000000000
000100010001000100000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000

%%%%%%%%
Codeword2
 pathological count= 10,cnt1= 11, cnt2= 2,  cnt3= 2, sum =25::
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000001
000000000000000000000000000000000000000000000000000000001000
000000000000000000000000000000000000000000000000000001000100
```

```
0000000000000000000000000000000000000000000000000000001000110001
0000000000000000000000000000000000000000000000000001000100000000

0010001100000101010100100001000001010000000000000000000000000000
0001000000000000000100000000000000000000000000000000000000000000
0101000000000000000000000000000000000000000000000000000000000000

%%%%%%%%
Codeword3
 pathological count= 6,cnt1 = 15, cnt2= 4,  cnt3= 0, sum = 25::
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000010
0000000000000000000000000000000000000000000000000000000000010000
0000000000000000000000000000000000000000000000000000000010000000
0000000000000000000000000000000000000000000000000000010000100000
0000000000000000000000000000000000000000000000000010000000000000

1010110011000110110010001010101000000000000000000000000000000000
0010001000100010000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
```

2.1 Acknowledgment

We thank Yiqun Lisa Yin for suggesting this problem.

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.

[CJ98] F. Chabaud and A. Joux. Differential collisions in SHA-0. In *Crypto*, 1998.

[JP05] Charanjit S. Jutla and Anindya C. Patthak. A Simple and Provably Good Code for SHA Message Expansion. Cryptology ePrint Archive, Report 2005/247, 2005. `http://eprint.iacr.org/`.

[MP05] K. Matusiewicz and J. Pieprzyk. Finding good differential patterns for attacks on SHA-1. In *International Workshop on Coding and Cryptography*, 2005.

[RO05] V. Rijmen and E. Oswald. Update on SHA-1. In *Lecture Notes in Computer Science, Vol. 3376, Springer*, 2005.

[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.

[WFLY04] X. Y. Wang, D. G. Feng, X. J. Lai, and H. B. Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. In *Short talk presented at CRYPTO'04 Rump Session and IACR eprint Archive, August*, 2004.

[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.