Restoring the Differential Resistance of MD6

Ethan Heilman

July 10, 2011

Abstract

These notes present new results to reestablish the differential resistance of MD6. In this paper we introduce a classification system of differential weight patterns that allows us to extend previous analysis to prove that MD6 is resistant to differential cryptanalysis. Our analysis allows us to more than double the security margin of MD6 against differential attacks.

Keywords: Differential Cryptanalysis, Cryptographic Hash Function, MD6, Computer Aided Proof

1 Introduction

These notes present new research on the security of the MD6 hash function against differential attacks. This work builds on ideas and methods from the MD6 SHA-3 NIST report[1], especially with regards to Section 6.9.1 on the differential security of MD6.

As noted in an Official Comment on MD6 posted to the NIST list, a gap was discovered in the proof that MD6 is resistant to differential attacks[2]. When this bug in the computer aided prover was patched, the prover was no longer capable of proving the differential security of MD6 within the recommended number of rounds for MD6. With access to the original prover source code, we researched additional techniques and improvements to the prover such that the differential security of MD6 could be firmly reestablished within the recommended number of round.

In this paper we develop an improved method of proving the differential resistance of MD6. The general approach we took was to determine what elements of the problem represent the greatest performance cost, find a trade-off that greatly reduces this performance cost for those elements, and use this trade-off to increase the lower bound by searching more rounds.

More specifically our approach is to classify differential weight patterns into two classes, trivial patterns and non-trivial patterns (definitions of these patterns of introduced in Section 4.1). In Section 4.1 we analyze why Non-trivial patterns represent a such significant performance cost. We reduce the performance cost of non-trivial patterns by finding an effective trade-off between speed and active AND gates in Section 4.2. This allows us to find a high lower bound for non-trivial patterns. Finally we assemble our improved non-trivial pattern lower bound and our trivial pattern lower bound into a combined lower bound that encompasses all differential weight patterns. This combined lower bound is then used, in Section 5, to reestablish the differential security of MD6 for all

recommended output sizes and rounds. Our combined lower bound proves that not only is MD6 resistant against differential cryptanalysis, but additionally MD6 has a security margin against differential cryptanalysis more than twice as large as originally claimed in the MD6 NIST Proposal[1].

2 Background

In Section 2.1 and Section 2.2 we briefly sketch out the relevant details of MD6 and differential cryptanalysis which we will need for our analysis.

2.1 MD6 Overview

As a complete description of MD6 is not necessary to our analysis we will instead provide a brief sketch covering the relevant details. A full description of MD6 can be found in the MD6 NIST Proposal[1].

Because the compression function of MD6 is a Feedback Shift Register we can conceptually view it as two interacting components: an array A of 64-bit words $A[0], A[1], A[2], ...A[n]$ and a feedback function which reads and writes to A.

The feedback function starts at index $i = 89$ (The words $A[0]$ through $A[88]$ contain the message which is being compressed). It uses six offsets called tap positions $t_0, t_1, t_2, t_3, t_4, t_5$ to index words $A[i - t_0], A[i - t_1], A[i - t_2], A[i - t_3], A[i - t_4], A[i - t_5]$ as input. Processing this input the feedback function generates a 64-bit output word which it writes to $A[i]$. Finally it increments i by 1 to move one word to the right. In this way the feedback function moves along A generating outputs until it reaches the last word of A. Each increment of i is the end of a step. Every 16 steps is the end of a round, every 89 steps is the end of a rotation. The value n is the number of words in A. The feedback function performs $n - 89$ steps to compute $n - 89$ values (the first 89 words of A are provided as input to the compression and thus are not computed).

For the purposes of our analysis we provide a simplified compression function. We have removed all features which do not have differential relevance and therefore have no bearing on our analysis:

Algorithm 1 COMPRESS(A[], ROUNDS)

$n \leftarrow ROUNDS * 16$
for $i = 89$ to n **do**
  $x \leftarrow feedback(A[i - t_0], A[i - t_1], A[i - t_2], A[i - t_3], A[i - t_4], A[i - t_5])$
  $A[i] \leftarrow x \oplus g(x)$
end for

The feedback function is given as:

$$feedback(I_0, I_1, I_2, I_3, I_4, I_5) : I_0 \oplus (I_1 \wedge I_2) \oplus (I_3 \wedge I_4) \oplus I_5.$$

The function $g(x)$ is given as

$$g(x) : (x >> R) \oplus (x << L)$$

where the values R are L determined by a table lookup indexed by i. As the full nature of the function $g(x)$ goes beyond the scope of this paper we will not be describing it in more detail.

2.2 Differential Cryptanalysis

Differential cryptanalysis[3] of a function reasons about the probability of particular differences in outputs given differences in inputs supplied to the function. The standard differential collision attack with a keyless cryptographic hash function is to find some difference in inputs to the hash function such that the resultant outputs have a relevantly high probability of having no differences.

As the compression function moves along the array A it propagates differences. We refer to a record of all the difference propagated through the compression as a differential path. At certain points the compression function either propagates or does not propagate a difference. This decision to propagate or not rests on inputs which can not be expressed wholly as differences. Therefore more than one differential path may satisfy some input difference. To show that a function is secure against differential attacks we show that any differential path has a probability of occurring less than $\frac{1}{2^{n/2}}$. Thus any interesting differential path is useless to an attacker because finding a real input which results in that path is as computationally costly as a brute force attack.

2.2.1 Terms

Active AND Gates (AAGs) are the places at which a differential path may split into two or more differential paths. They are decision points in which a number of particular differences are either propagated or not depending on values which are not specified by differences. For example, a difference may specify that one bit is the same as another bit but a difference can not specify the actual value of the bits, that the bits are both zero or that the bits are both one. To explain AAGs we provide the following demonstration.

Consider four values, $A, B, A', B' \in \{1, 0\}$ supplied to an AND gate in pairs $(A \wedge B)$ and $(A' \wedge B')$. The difference of their outputs is

$$Z = (A \wedge B) \oplus (A' \wedge B').$$

We can calculate the probability of Z having a particular value given information about the equality of A and A', and B and B'. Knowing only that A is not equal to A' and B is equal to B' we can not say the value of the result Z. Given that $A \neq B$ if we further assume $B = 1$ and $B' = 1$ then we know that $Z = 1$ Since in our differential analysis we can't know the exact value of B but only the difference between B and B', we can only calculate the probability that $Z = 1$.

This probability can be broken into two cases:

1. If $A = A'$ and $B = B'$ then the prob$(Z = 0)$ is 1.
2. If $A \neq A'$ or $B \neq B'$ then the prob$(Z = 0) = 0.5$ and prob$(Z = 1)$ is 0.5.

The probability that any AND gate will pass on a difference is given by the probability that Z will be 1. We say that an AND gate within MD6 is active when it is computing a difference.

Differential Weight Patterns are patterns of difference in the array A in the MD6 compression function. The difference measured in a differential weight pattern is the Hamming weight difference. For example the differential weight pattern $[3.1, 5.32]$ corresponds to a Hamming weight of 1 at position $A[3]$ and Hamming weight of 32 at position $A[5]$. For a more in depth discussion of differential weight patterns see the MD6 NIST Proposal[1].

Differential Positions are positions in a differential weight pattern for which the exact magnitude of that difference is indeterminate but non-zero. If we have two difference weight patterns such as $[17.2, 23.5, 50.1]$ and $[17.1, 18.1, 50.4]$, we can say that both patterns share a differential position at 17 and 50. The main difference between differential position patterns and differential weight patterns is that position patterns don't specify a magnitude. For example we write a differential position pattern like $[2, 5, 7, 14]$.

3 Prior Work on MD6 Differential Resistance

We will briefly summarize the prior differential resistance work presented in the MD6 NIST Proposal. For full details and assumptions underlying this proof read the MD6 NIST Proposal[1].

To prove the resistance of MD6 to differential attacks, one must show that the lower bound on the workload of a differential attack exceeds the workload of the birthday attack. This workload lower bound can be found by computing a lower bound on the minimum number of AND gates activated by all valid differential weight patterns given the randomness and independence assumption made and justified in Section 6.9.1 of the MD6 NIST Proposal[1]. Under this assumption each Active AND Gate (AAG) increases the workload to find a collision for a particular path by a factor of 2 (the workload for n AAGs is 2^n). The birthday attack for an n-bit hash function requires $2^{n/2}$ computations, thus if a differential path contains at least $n/2$ AAGs, the workload of a differential attack must be at least as large as the birthday attack. The proof presented in the MD6 NIST Proposal succeeds if it can show that the number of AAGs is 256 (512 bit hash input) or higher for r rounds, where $r = (168 - 15) = 153$. [1]

3.1 Lower Bounding Segments

As the number of all possible differential paths is quite high a computer is necessary to search through all the possibilities. Even with a computer the number of possible differential paths is so large as to be impractical to check.

$$AAG_r \geq AAG_s \times \lfloor r/s \rfloor \tag{6.8}$$

Rather than checking all r rounds we can find the AAG lower bound in s rounds (for a small s) and using this lower bound and eq. 6.8[1], extend the smaller s round AAG lower bound (AAG_s) to a full r round lower bound (AAG_r). As shown in Algorithm 2, we find the AAG lower bound on progressively larger segments until the computation becomes infeasible.

[1] While the recommended number of rounds for MD6 is 168 rounds, the NIST MD6 proposal assumes a security margin of 15. Leaving the proof 153 rounds to work with.

Algorithm 2 LowerBound_Segments()

```
while s ≤ 153 do
    while maxAAG ≤ 256 do
        pathFound ← search(s, maxAAG)
        if pathFound then
            s ← s + 1
        else
            maxAAG ← maxAAG + 1
            print s, maxAAG
        end if
    end while
end while
```

To lower bound s rounds we search through all valid paths of length s which activate no more than $maxAAG$ AND gates (searching all the paths that do no exceed the upper bound of $maxAAG$). If no such paths exist then we have proved a lower bound $maxAAG$ of AND gates activated for all possible differential paths of s rounds or less.

3.2 Counting AAGs

As the exact rules we use for counting AAGs will be will be important to our improved results we introduce them here. For reasons of efficiency we are not dealing with exact differences but rather with differential weight patterns. Thus the question becomes given four hamming differences H_{I_1}, H_{I_2}, H_{I_3}, and H_{I_4}, corresponding to four of the six inputs to the feedback function, how many AND gates are activated. The number of AND gates activated is equivalent to the difference weight of the four hamming differences with three exceptions:

Three Exceptions of Counting Active AND gates:

1. **Any differences that share the same AND gate only count once.**

   As we are computing a lower bound on the number of AND gates activated we must assume that the differences align to optimally reduce the number of AAGs counted. Thus the lower bound on the number of AND gates activated by either H_{I_1} and H_{I_2} or H_{I_3} and H_{I_4} is:

   $$AAG_{I_1 I_2} = \max(H_{I_1}, H_{I_2})$$

   $$AAG_{I_3 I_4} = \max(H_{I_3}, H_{I_4})$$

2. **Any differences that share the same step only count once.**

   Two sets of AND gates can be activated in one step because the MD6 feedback function (see Section 2.1) contains two sets of AND gates $(I_1 \wedge I_2) \oplus (I_3 \wedge I_4)$. Only one output word is generated therefore allowing for the possibility that two activated AND gates will align to the same output bit. The lower bound on the number of AND gates activated for the output word of a step is the max of the two sets.

   $$AAG_{new} = \max(AAG_{I_1 I_2}, AAG_{I_3 I_4})$$

3. **AND gates activated when one of the inputs to an AND gate falls off the beginning of a segment are ignored** To avoid double counting we ignore AND gates that are activated when one of the inputs (tap positions) to an AND gate falls off the beginning of a segment.[2] As we do not count AND gates activated by inputs that fall off the beginning of a segment, we are free to count AND gates activated by inputs that fall off the end of a segment without fear of double counting.

Using these these rules we develop an algorithm for counting AAGs, shown as Algorithm 3. Note that the parameters to the Count_AAGS function are the Differential Weight Pattern we are counting denoted as DWP, and four integers, t_1, t_2, t_3 and t_4, that store the tap positions (see Section 2.1). We start counting AAGs at position t_4 since all differences where $i < t_4$ would not count due to exception 3.

Algorithm 3 COUNT_AAGS(DWP[ ], t_1, t_2, t_3, t_4)

$AAG = 0$
for $i = t_4$ to $(len(DWP) + t_4)$ **do**
  $AAG_{t_1} \leftarrow 0, AAG_{t_2} \leftarrow 0, AAG_{t_3} \leftarrow 0, AAG_{t_4} \leftarrow 0$
  if $(i - t_1) < len(DWP)$ **then**
    $AAG_{t_1} \leftarrow DWP[i - t_1]$
  end if
  if $(i - t_2) < len(DWP)$ **then**
    $AAG_{t_2} \leftarrow DWP[i - t_2]$
  end if
  if $(i - t_3) < len(DWP)$ **then**
    $AAG_{t_3} \leftarrow DWP[i - t_3]$
  end if
  $AAG_{t_4} \leftarrow DWP[i - t_4]$
  $AAG = AAG + \max(AAG_{t_1}, AAG_{t_2}, AAG_{t_3}, AAG_{t_4})$
end for
return AAG

As noted in an Official Comment on MD6 posted to the NIST list, a gap was discovered in the differential resistance proof[2]. The gap consisted of a bug in the program that searched through all valid paths to establish the AAG lower bound for s rounds. When this bug was corrected by members of the MD6 team, the lower bounds generated (see Table 1) were such that 153 rounds were no longer sufficient to prove that at least 256 AND gates are activated and therefore no longer sufficient to prove differential resistance for 512-bit MD6. The limiting factor in the proof was not the proof method per se, but rather the performance characteristics of the program that was searching these paths. The number of possible paths increases dramatically with higher values of s and this increase makes finding a sufficient lower bound computationally infeasible.

We will now, in detail, examine the results and implications from the corrected prover. According to the results from the corrected prover, as presented

[2]For example consider the case in which the variable i from Algorithm 1 is small enough that $i - t_4 < 0$. Under these circumstances the tap position t_4 actually exists on the previous segment and is therefore undetermined.

in Table 1, a lower bound of 256 AAGs requires at least 244 rounds (not including the security margin of 15 rounds).

We will now walk through the steps we took to extend the results in Table 1 to the result for 244 rounds. First, given the lower bound for 14 rounds of 15 AAGs we use eq 6.8 to extend this lower bound to 244 rounds. Then, repeating the lower bound for 14 rounds 17 times, we get a lower bound of 255 AAGs for 238 rounds.

$$17 \times 14 = 238 \text{ rounds}$$

$$255 \text{ AAGs} = 15 \times \lfloor 238/14 \rfloor$$

255 AAGs is just shy of the 256 AAGs we require, so we increase the number of rounds by 6, and use the lower bound for 6 rounds to get add an additional 2 AAGs.

$$17 \times 14 + 6 = 244 \text{ rounds}$$

$$257 \text{ AAGs} = (15 \times \lfloor 238/14 \rfloor) + (2 \times \lfloor 6/6 \rfloor)$$

The MD6 NIST Proposal assumed an additional security margin of 15 rounds, raising the number of rounds to 259. As the recommended number of rounds for 512 bit MD6 only is 168 rounds, the corrected program is unable to show that 512 bit MD6 is secure against differential attacks within the recommended number of rounds.

s	6–8	9–10	11	12	13	14
LB on AAG_s	2	4	5	11	13	$\geq$15

Table 1: Computer-generated AAG lower bounds for s-round differential weight patterns, using the corrected initial program.

4 Computing an Improved Lower Bound

In Section 3, we discussed a method for lower bounding the workload of a standard differential attack on MD6. Unfortunately the computer generated element of the proof was too computationally intensive to prove a satisfactory lower bound. In this section we build on this prior method to show an improved method which provides a satisfying lower bound.

We do this in three steps. First in Section 4.1 we classify differential weight patterns into two classes, trivial and non-trivial patterns. We use the simple nature of trivial patterns to quickly compute a trivial patterns lower bound. Second in Section 4.2 we reason about the properties of non-trivial patterns to develop a more cost effective strategy for lower bounding non-trivial patterns which we use to lower bound non-trivial patterns. Third and finally in Section 4.3 we assemble the trivial and non-trivial lower bound into a general lower bound.

4.1 Types of Differential Weight Patterns

We classify differential weight patterns into two classes: trivial and non-trivial. In essence we are attempting to classify patterns that are cheap to lower bound and patterns that are expensive to lower bound, allowing us to focus a cost-benefit analysis on the expensive patterns (non-trivial patterns). Our main intuition in lower bounding non-trivial patterns is that the computational cost of finding AAGs varies depending on what part of the pattern we are searching. We save computational resources by not searching for AAGs on the expensive parts of a pattern, which allows us to spend these resources with far greater effect on the more affordable sections of a pattern.

A Trivial pattern is any differential weight pattern which has three or less differential positions in its first rotation.

A Non-trivial pattern is any differential weight pattern that contains at least four differential positions in its first rotation, and at least four differential positions in each subsequent rotation.

Here we give some examples of trivial and non-trivial patterns :

Trivial	$[71.4]$
	$[16.2, 54.1, 88.2]$
	$[14.2, 35.1, 86.2, 124.2, 141.2, 158.2, 213.2, 230.2, 302.2]$
Non-Trivial	$[1.1, 2.1, 3.1, 4.1]$
	$[32.1, 35.1, 45.1, 81.1, 91.2]$
	$[17.1, 35.1, 45.1, 81.1, 147.2, 150.2, 157.2, 160.2]$

Note that any rotation in MD6 must either start a trivial pattern or a non-trivial pattern, as any pattern within a rotation must either have less than 3 differential positions or greater than 3 positions. Which is to say that for the first rotation trivial and non-trivial patterns are mutually exclusive. Also note that a trivial pattern can contain a non-trivial pattern in its second or subsequent rotation, just not in its first rotation.

Trivial patterns are patterns that are computationally cheap to compute as the restriction on the number of possible positions in the first rotation greatly reduces the number of possible valid positions for the first rotation and subsequent rotations. Consider searching all the valid differential weight patterns under a AAGs. There are at most

$$\sum_{k=1}^{a} \binom{89}{k}$$

differential positions in the first rotation. But for a trivial pattern we are restricted to three or less differential positions and thus there are at most

$$\sum_{k=1}^{3} \binom{89}{k} = \binom{89}{3} + \binom{89}{2} + 89 = 117,569$$

possible patterns for the first round.

Just as our AAG lower bound on trivial patterns is inexpensive to compute, the lower bound on non-trivial patterns is expensive to compute (we chose the definitions that we did for trivial and non-trivial patterns precisely because it was a definition that broke lower bounding the differential weight patterns into

an easy and a hard category). Non-trivial patterns have at least four differential positions in each rotation, thus a non-trivial pattern costs at least

$$\binom{89}{4} = 2,441,626$$

Since we need to search well beyond the first rotation, the above analysis is not complete but it serves as a point of reference for our intuition. The more differential positions in the first rotation, the more possible differential positions in subsequent rotations. Thus, not only will trivial patterns have a smaller number of possible patterns in the first rotation, they are also likely to have fewer possible patterns to search in each subsequent rotation. Practically this means that the number of possible trivial patterns is small enough that we can compute the lower bound for trivial patterns using only the original approach from the MD6 report.

4.2 Computing Non-Trivial Patterns

Our technique for finding the AAG lower bound for non-trivial patterns is to trade off the opportunity to count some AAGs for the ability to search more rounds. This trade-off rests on the following observation; differences in the first 48 steps of the first rotation have a lower limit on the number of AND gates they can activate.

As explained in exception 3 in Section 3.2, we avoid potential over counting of AAGs by ignoring AAGs that might have been counted on a previous segment. A differential weight pattern which occurs in the first rotation can result in significantly less AAGs than if that very same pattern had occurred in a subsequent rotation. For example a difference occurring in word 2 of a segment can only activate a maximum of AND gates equal to 1 times the weight of the difference whereas a difference occurring in word 71 can activate a maximum of AND gates 4 times the weight of its difference[3]. This fundamental inequality of locations is the heart of our plan to improve the computational cost per AAG for non-trivial patterns.

Our strategy is to skip first rotation and instead start the search on the second rotation. Since we know that the first rotation has at least 4 differential positions, we add an additional 4 AAGs to our total lower bound. While this means that we will likely undercount the first rotation, doing so opens up more fruitful rounds that previously were computationally beyond our reach. For a purely computational standpoint trying all possible positions in the second rotation is as complex as trying all possible positions the first rotation, but searching the second rotation results in more AND gates being activated earlier in the search and therefore allows us to hit our AAG goals quicker. In practice this means that we run the prover introduced in the NIST MD6 Report with a simple modification such that it skips the first rotation, adds an additional 4 AAGs to the final result, and only computes patterns with at least 4 differential patterns per rotation.

[3]The reader may wish to further explore and verify this observation by running some sample differential weight patterns containing only a single difference as input to Algorithm 3 and noting how the position in which a difference occurs can alter the maximum number of AND gates that a difference can activate.

Rounds	Trivial LB on AAG_s	Non-Trivial LB on AAG_s
6	2	4
7	2	4
8	2	6
9	4	7
10	4	7
11	5	7
12	11	14
13	13	16
14	19	16
15	22	17
16	22	≥ 18
17	30	≥ 35
18	34	–
19	35	–
20	38	–
21	39	–
22	47	–
23	55	–
24	60	–

Table 2: The Trivial and Non-Trivial Differential Weight Patterns AAG Lower Bound

4.3 Assembling Trivial and Non-Trivial Patterns

After computing an AAG lower bound for trivial and non-trivial patterns, we need to assemble these lower bounds into a lower bound that exceeds 256 AAGs for some number of rounds r. We do this in three steps: first we build a list of all possible concatenations of the trivial and non-trivial lower bounds that are at least r (where r is the number of rounds we allow ourselves for MD6), second we compute the AAG count for each element of the list and finally we choose the element in the list that has the minimum AAGs count. This minimum AAG combination is our extended lower bound. As the process for assembling these lower bounds is fairly straightforward, we will not go into it in more detail in this paper but interested parties may wish to consult the source code published alongside this paper (see Section 6).

5 Results

Running the improved algorithm discussed in Section 4.1 we get the results for Trivial and Non-Trivial patterns shown in Table 2. Using the method outlined in 4.3, we assemble the trivial and non-trivial lower bounds to find an extended lower bound for all the recommended input sizes of MD6, shown in Table 3.

We get a lower bound of 257 AAGs for 137 rounds for 512 bit MD6. Including the 15 round security margin, we get 152 rounds which is well below the recommended 168 rounds. Therefore, not only is the number of rounds (168 rounds) used by MD6 sufficient to prove differential security, but a reduced

input size	resistant rounds	MD6 rounds	security margin	AAGs
40 bits	24	50	26	25
80 bits	29	60	31	41
128 bits	43	72	29	68
160 bits	52	80	28	82
224 bits	66	96	30	113
256 bits	76	104	28	128
384 bits	106	136	30	192
512 bits	137	168	31	257

Table 3: Our complete results showing: the number of rounds necessary to prove differential resistance for each of the proposed input sizes (resistant rounds), the number of rounds recommended by the MD6 team (MD6 rounds), and the new security margin our results establish.

round 512-bit MD6 could make the same guarantee allowing us to more than double the security margin from 15 to 31 rounds. As discussed in Section 3.2 the original technique required 259 rounds to show that 512-bit MD6 was resistant; the new method results in a round reduction of 107 rounds.

6 Source Code

As the original result of the differential resistant of MD6 was overturned due to a bug in the computer aided prover we throughly investigated the code, analyzed the output for errors and wrote a series of exhaustive tests to increase our confidence in the correctness of the prover. These tests are also made available in the source code package.

We would like to note that much of the code remains unchanged from the program that was initially used in the MD6 differential resistance proof presented in the MD6 report. We built on the works of others.

7 Conclusion

In this paper we discussed an improved method for proving the differential security of MD6 by introducing the classes of differential patterns, and investigating the properties of these classes to extend the performance envelope of the computer aided proof. The results from this method reestablished MD6's security claims by proving MD6's resistance to differential cryptanalysis. Additionally we show that a reduced 152 round variant of MD6 would also be secure and that the recommended number of rounds for MD6 (168 rounds) has a security margin of more than twice as large as originally claimed.

8 Acknowledgments

We would like to ackowledge all the members of the MD6 team (Ron Rivest, Benjamin Agre, Dan Bailey, Sarah Cheng, Christopher Crutchfield, Yevgeniy

Dodis, Kermin Fleming, Asif Khan, Jayant Krishnamurthy, Yuncheng Lin, Leo Reyzin, Emily Shen, Jim Sukha, Eran Tromer, Yiqun Lisa Yin) for their work on the original differential proof to which this paper owes an enormous intellectual debt, especially Emily Shen and Yiqun Lisa for both providing help and support, and Ron Rivest for providing advice and getting me involved in this topic in the first place.

References

[1] Ronald L. Rivest, Benjamin Agre, Daniel V. Bailey, Christopher Crutchfield, Yevgeniy Dodis, Kermin Elliott Fleming, Asif Khan, Jayant Krishnamurthy, Yuncheng Lin, Leo Reyzin, Emily Shen, Jim Sukha, Drew Sutherland, Eran Tromer, Yiqun Lisa Yin. The MD6 hash function - A proposal to NIST for SHA-3. Submission to NIST - `http://groups.csail.mit.edu/cis/md6/docs/2009-04-15-md6-report.pdf`, 2009.

[2] Ronald L. Rivest. OFFICIAL COMMENT: MD6. NIST mailing list (`http://groups.csail.mit.edu/cis/md6/OFFICIAL_COMMENT_MD6_2009-07-01.txt`), 2009.

[3] E. Biham and A. Shamir. Differential cryptanalysis of des-like cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.