

# Pseudo-cryptanalysis of Blue Midnight Wish

Søren S. Thomsen  
crypto@znoren.dk

April 16, 2009

## Abstract

We describe pseudo-collision and pseudo-(second) preimage attacks on the SHA-3 candidate Blue Midnight Wish. The complexity of the pseudo-collision attack is around  $2^{3n/8+1}$ , and the complexity of the pseudo-(second) preimage attack is around  $2^{3n/4+1}$ .

## 1 Introduction

Blue Midnight Wish [3] is a cryptographic hash function submitted to the SHA-3 competition [4]. In this note we describe how to find preimages and collisions in the compression function of Blue Midnight Wish faster than expected. The results are independent of the security parameter of Blue Midnight Wish. Previous cryptanalytic results on Blue Midnight Wish are limited to near-collisions in the compression function [6].

We start off with a brief description of Blue Midnight Wish, and then we describe how to carry out the attacks on the compression function, which can also be seen as pseudo-attacks.

## 2 Description of Blue Midnight Wish

Blue Midnight Wish (BMW) is a design that applies only four types of operations: modular additions, XORs, shifts, and rotations. In the following, unless stated otherwise, all additions are to be taken modulo  $2^w$ , where  $w$  is the word size. The word size is 32 bits for BMW-256, and 64 bits for BMW-512. In this paper we consider only these two variants of BMW. We now give a brief description of Blue Midnight Wish – for further details, we refer to the specification [3].

Blue Midnight Wish maintains a state that is twice as large as the digest size. A message to be hashed is padded by appending a ‘1’-bit, then a number of ‘0’-bits, and finally a 64-bit representation of the original message length in bits. The number of ‘0’-bits is chosen such that the padded message has a length that is a multiple of  $16w$  bits. The message is then split into blocks of  $16w$  bits, and each block is processed by a compression function, iterated in Merkle-Damgård mode, with the extension that the last compression function output is truncated down to the digest size  $n$  by taking the  $n$  least significant bits.

The Blue Midnight Wish compression function consists of three sub-functions  $f_i$ ,  $i \in \{0, 1, 2\}$ . The compression function takes 16 message words and 16 chaining words of input. Let  $M = M_0 \parallel \dots \parallel M_{15}$  denote the message block, and let  $H = H_0 \parallel \dots \parallel H_{15}$  denote the chaining input. The function  $f_0$  is a permutation with input  $M \oplus H$ . Denote by  $Q_0 \parallel \dots \parallel Q_{15}$  the output words of  $f_0$ . The function  $f_1$  is a multi-permutation with inputs  $M$  and  $Q_i$ ,  $i = 0, \dots, 15$ . Denote by  $Q_{16} \parallel \dots \parallel Q_{31}$  the output words of  $f_1$ . Compression takes place in the function  $f_2$ , which takes as input  $M$  and  $Q_i$ ,  $i = 0, \dots, 31$ , and returns a new chaining variable which we denote by  $H^* = H_0^* \parallel \dots \parallel H_{15}^*$ .

In attacks on the compression function,  $f_0$  can be ignored, since given  $M$  and  $Q_i$ ,  $i = 0, \dots, 15$ , the chaining input can be computed as  $H = f_0^{-1}(Q_0 \| \dots \| Q_{15}) \oplus M$ . Hence, we do not describe  $f_0$  in detail here.

Since  $f_1$  is a multi-permutation,  $f_1(M, Q_0 \| \dots \| Q_{15})$  is a permutation when  $M$  is fixed, and also when the  $Q_i$  are fixed. This also means that given  $Q_i$ ,  $i = 0, \dots, 31$ , the input  $M$  can be computed, and given  $M$  and  $Q_i$ ,  $i = 16, \dots, 31$ , the words  $Q_i$ ,  $i = 0, \dots, 15$  can be computed.

## 2.1 Details of $f_1$

The definition of  $f_1$  depends on the choice of a security parameter  $P \leq 16$ , which selects the number of “complex” rounds. The number of “simple” rounds is  $16 - P$ . The default number of complex rounds is 2, and we use this value here. However, as mentioned in the introduction, our results would not be affected by a different choice of  $P$ .

A number of permutations  $s_i$ ,  $0 \leq i \leq 5$  and  $r_i$ ,  $1 \leq i \leq 7$ , are defined. All these permutations can easily be inverted, and we omit a description of the details.

A complex round computes  $Q_i$  based on  $Q_{i-1}, \dots, Q_{i-16}$ ,  $16 \leq i < 16 + P$  as follows.

$$\begin{aligned} Q_i \leftarrow & s_1(Q_{i-16}) + s_2(Q_{i-15}) + s_3(Q_{i-14}) + s_0(Q_{i-13}) \\ & + s_1(Q_{i-12}) + s_2(Q_{i-11}) + s_3(Q_{i-10}) + s_0(Q_{i-9}) \\ & + s_1(Q_{i-8}) + s_2(Q_{i-7}) + s_3(Q_{i-6}) + s_0(Q_{i-5}) \\ & + s_1(Q_{i-4}) + s_2(Q_{i-3}) + s_3(Q_{i-2}) + s_0(Q_{i-1}) \\ & + W_{i-16} + K_{i-16}. \end{aligned}$$

Here, all  $K_{i-16}$  are constants, and  $W_{i-16} = M_{i-16} + M_{i-13} - M_{i-6}$ , where indices are to be taken modulo 16. The mapping from  $(M_0, \dots, M_{15})$  to  $(W_0, \dots, W_{15})$  can be seen as a multiplication modulo  $2^w$  with the circulant matrix  $\mathbf{B}$  having in its first row the elements

$$(1, 0, 0, 1, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0).$$

This matrix is invertible modulo  $2^w$ . The inverse matrix  $\mathbf{B}^{-1}$  is (naturally) also circulant, and for  $w = 32$  the first row contains the elements

$$\begin{aligned} & (\text{abababac, c6c6c6c7, bdbdbdbe, c0c0c0c1, 15151515, 4e4e4e4e, 90909090, fcfcfd0,} \\ & \text{babababb, 6c6c6c6d, dbdbdbdc, 0c0c0c0c, 51515151, e4e4e4e5, 09090909, fcfcfd}) \end{aligned}$$

in hexadecimal. For  $w = 64$ , the words in the inverse matrix can be found by a predictable expansion of the words above.

A simple round computes  $Q_i$  based on  $Q_{i-1}, \dots, Q_{i-16}$ ,  $16 + P \leq i < 32$  as follows.

$$\begin{aligned} Q_i \leftarrow & Q_{i-16} + r_1(Q_{i-15}) + Q_{i-14} + r_2(Q_{i-13}) \\ & + Q_{i-12} + r_3(Q_{i-11}) + Q_{i-10} + r_4(Q_{i-9}) \\ & + Q_{i-8} + r_5(Q_{i-7}) + Q_{i-6} + r_6(Q_{i-5}) \\ & + Q_{i-4} + r_7(Q_{i-3}) + s_5(Q_{i-2}) + s_4(Q_{i-1}) \\ & + W_{i-16} + K_{i-16}. \end{aligned}$$

It is clear that in the case of both a complex and a simple round, e.g.,  $Q_{i-16}$  may be computed from  $Q_i, \dots, Q_{i-15}$  and  $W_{i-16}$ .

## 2.2 Details of $f_2$

In  $f_2$ , the 48 words  $M_i$ ,  $0 \leq i \leq 15$ , and  $Q_i$ ,  $0 \leq i \leq 31$ , are compressed down to 16 words  $H_i^*$ ,  $0 \leq i \leq 15$ .

First, two temporary values  $XL$  and  $XH$  are computed as

$$\begin{aligned} XL &= Q_{16} \oplus Q_{17} \oplus \dots \oplus Q_{23} \\ XH &= XL \oplus Q_{24} \oplus Q_{25} \oplus \dots \oplus Q_{31}. \end{aligned}$$

Then, the  $H_i^*$  are computed as

$$\begin{aligned} H_0^* &= (XH^{\ll 5} \oplus Q_{16}^{\gg 5} \oplus M_0) + (XL \oplus Q_{24} \oplus Q_0) \\ H_1^* &= (XH^{\gg 7} \oplus Q_{17}^{\ll 8} \oplus M_1) + (XL \oplus Q_{25} \oplus Q_1) \\ H_2^* &= (XH^{\gg 5} \oplus Q_{18}^{\ll 5} \oplus M_2) + (XL \oplus Q_{26} \oplus Q_2) \\ H_3^* &= (XH^{\gg 1} \oplus Q_{19}^{\ll 5} \oplus M_3) + (XL \oplus Q_{27} \oplus Q_3) \\ H_4^* &= (XH^{\gg 3} \oplus Q_{20} \oplus M_4) + (XL \oplus Q_{28} \oplus Q_4) \\ H_5^* &= (XH^{\ll 6} \oplus Q_{21}^{\gg 6} \oplus M_5) + (XL \oplus Q_{29} \oplus Q_5) \\ H_6^* &= (XH^{\gg 4} \oplus Q_{22}^{\ll 6} \oplus M_6) + (XL \oplus Q_{30} \oplus Q_6) \\ H_7^* &= (XH^{\gg 11} \oplus Q_{23}^{\ll 2} \oplus M_7) + (XL \oplus Q_{31} \oplus Q_7) \\ H_8^* &= (H_4^*)^{\ll 9} + (XH \oplus Q_{24} \oplus M_8) + (XL^{\ll 8} \oplus Q_{23} \oplus Q_8) \\ H_9^* &= (H_5^*)^{\ll 10} + (XH \oplus Q_{25} \oplus M_9) + (XL^{\gg 6} \oplus Q_{16} \oplus Q_9) \\ H_{10}^* &= (H_6^*)^{\ll 11} + (XH \oplus Q_{26} \oplus M_{10}) + (XL^{\ll 6} \oplus Q_{17} \oplus Q_{10}) \\ H_{11}^* &= (H_7^*)^{\ll 12} + (XH \oplus Q_{27} \oplus M_{11}) + (XL^{\ll 4} \oplus Q_{18} \oplus Q_{11}) \\ H_{12}^* &= (H_0^*)^{\ll 13} + (XH \oplus Q_{28} \oplus M_{12}) + (XL^{\gg 3} \oplus Q_{19} \oplus Q_{12}) \\ H_{13}^* &= (H_1^*)^{\ll 14} + (XH \oplus Q_{29} \oplus M_{13}) + (XL^{\gg 4} \oplus Q_{20} \oplus Q_{13}) \\ H_{14}^* &= (H_2^*)^{\ll 15} + (XH \oplus Q_{30} \oplus M_{14}) + (XL^{\gg 7} \oplus Q_{21} \oplus Q_{14}) \\ H_{15}^* &= (H_3^*)^{\ll 16} + (XH \oplus Q_{31} \oplus M_{15}) + (XL^{\gg 2} \oplus Q_{22} \oplus Q_{15}). \end{aligned}$$

Here,  $x^{\ll r}$  means  $x$  left-shifted by  $r$  positions,  $x^{\gg r}$  means  $x$  right-shifted by  $r$  positions, and  $x^{\lll r}$  means  $x$  rotated left by  $r$  positions.

### 3 The attacks

The idea of the attacks is to fix  $Q_i$ ,  $i = 16, \dots, 31$  to a certain value, which means that the function  $f_2$  becomes rather simple. If, for instance,  $Q_i = 0$  for  $16 \leq i < 32$ , then one gets

$$\begin{aligned}
H_0^* &= M_0 + Q_0 \\
&\vdots \\
H_8^* &= (H_4^*)^{\lll 9} + M_8 + Q_8 \\
H_9^* &= (H_5^*)^{\lll 10} + M_9 + Q_9 \\
H_{10}^* &= (H_6^*)^{\lll 11} + M_{10} + Q_{10} \\
H_{11}^* &= (H_7^*)^{\lll 12} + M_{11} + Q_{11} \\
H_{12}^* &= (H_0^*)^{\lll 13} + M_{12} + Q_{12} \\
H_{13}^* &= (H_1^*)^{\lll 14} + M_{13} + Q_{13} \\
H_{14}^* &= (H_2^*)^{\lll 15} + M_{14} + Q_{14} \\
H_{15}^* &= (H_3^*)^{\lll 16} + M_{15} + Q_{15}
\end{aligned}$$

As an example, one may fix  $Q_i = 0$  for  $1 \leq i < 32$ , and vary only  $Q_0$ . Due to the “message schedule” in  $f_1$  (the matrix multiplication with  $\mathbf{B}$ ), all message words  $M_i$  may be affected by varying  $Q_0$ , when  $M_i$  are computed from  $f_1$ . This would cause all output words to be affected. However, if one tries two different values of  $Q_0$  such that they only differ in the most significant bit after the application of  $s_1$ , then only the most significant bits of some of the words  $M_i$  will be flipped. This results in a near-collision (a similar technique was used in [6]).

By using a slightly more sophisticated technique, some output message words can be set to any value chosen by the attacker. We now describe this technique.

#### 3.1 Controlling two output words

In the remainder of this section we assume that  $Q_i = 0$  for all  $i$ ,  $16 \leq i \leq 31$ . These words of  $Q$  could be fixed to other values, and the same attacks would apply, but fixing them to zero simplifies things a little.

The words  $M_i$  and the words  $W_i$  are related through the matrix  $\mathbf{B}$ . There is enough freedom in the choice of message words to ensure that, e.g., the words  $M_7$  and  $M_{11}$  are fixed to some constants chosen by the attacker, and at the same time control the words  $Q_7$  and  $Q_{11}$  via the words  $W_7$  and  $W_{11}$ . This enables the attacker to completely control the output words  $H_7^*$  and  $H_{11}^*$ .

As an example, we consider BMW-256 (i.e., the word size is  $w = 32$ ). We fix  $M_{13}$  to be equal to the padding word 00000080, and we fix  $M_{14}$  and  $M_{15}$  so that they contain correct length padding for a message of length  $512 - 96 = 416$  bits. We also fix  $M_7$  and  $M_{11}$  to arbitrary values. If we can ensure that  $Q_7 = \alpha - M_7$  and  $Q_{11} = \beta - \alpha^{\lll 12} - M_{11}$ , then we see that  $H_7^* = \alpha$  and  $H_{11}^* = \beta$ . As mentioned, controlling  $Q_7$  and  $Q_{11}$  can be done directly via the words  $W_7$  and  $W_{11}$ .

After fixing the mentioned words of  $M$  and keeping the remaining words free, we compute  $W = \mathbf{B} \cdot M$ . This vector contains elements that in some way depend on the message words  $M_i$ , but we may revert this dependency by a careful choice of the free message words  $M_i$ , such that some of the words  $W_i$  are free instead. For instance, we have that  $W_{15} = M_2 - M_9 + M_{15}$ . We may now replace  $M_2$  by  $W_{15} + M_9 - M_{15}$  throughout the vector, and get  $W_{15}$  as a free parameter. We use this method to make  $W_i$  free for all  $i$  down to 5 (recall that five of the words

$M_i$  are fixed). We now have that  $W$  is the vector

$$\begin{bmatrix} 3M_7 + M_{11} - 2M_{13} + 2M_{14} - 2M_{15} + W_6 - W_7 - W_9 - W_{11} + W_{12} + 2W_{13} - W_{14} - W_5 \\ -M_7 - 3M_{11} + M_{13} - 4M_{14} + 2M_{15} + 2W_5 + W_7 - W_{10} + 2W_{11} + 2W_{14} \\ -M_7 + M_{11} + M_{13} - M_{14} + 4M_{15} + 2W_5 - W_6 - 2W_8 + W_{11} - W_{12} - W_{13} - W_{15} \\ 2M_7 - 3M_{13} + M_{14} - 3M_{15} - W_5 + 2W_6 + W_8 - W_9 - W_{11} + W_{12} + 2W_{13} + W_{15} \\ -M_{11} + M_{13} - 3M_{14} + M_{15} + W_5 + W_7 - W_{10} + W_{11} + W_{14} \\ W_5 \\ \vdots \\ W_{15} \end{bmatrix}$$

For any choice of the words  $W_i$ ,  $5 \leq i < 16$ , we have that  $M = \mathbf{B}^{-1}W$  has the right values in positions 7, 11, 13, 14, and 15 when  $W_i$ ,  $0 \leq i \leq 4$ , are chosen according to the expressions above. We can now compute  $Q_i$ ,  $0 \leq i < 16$ , in the backward direction, but for  $i \in \{7, 11\}$ , we fix  $Q_i$  and compute the value of  $W_i$  needed to obtain the right values of  $Q_i$ . Note that  $Q_1$  and  $Q_0$  must be computed by inverting a ‘‘complex’’ round, and the rest of the  $Q_i$  are computed by inverting a ‘‘simple’’ round.

Nine of the words  $W_i$  can be chosen freely, and hence we may compute  $2^{9 \cdot 32} = 2^{288}$  different outputs of the compression function, for which the words  $H_7^*$  and  $H_{11}^*$  are fixed to  $\alpha$  and  $\beta$ , respectively. Note that only  $H_{11}^*$  ‘‘survives’’ the final truncation. Since the resulting message block is a correctly padded message, this technique can be used to mount pseudo-attacks on Blue Midnight Wish.

The complexity of a pseudo-collision attack (with different initial values for the colliding messages) is thus reduced to  $2^{7n/16}$ , and the complexity of a pseudo-preimage (and pseudo-second preimage) attack is reduced to  $2^{7n/8}$ .

Note that the memory requirements of the attacks are negligible (in the case of the pseudo-collision attack, cycle-finding methods such as [1, 2, 5] may be used).

### 3.2 Extending to four output words

The technique described in the previous subsection can be extended such that the attacker has control over four output words of the compression function, of which two are among the last eight words and therefore ‘‘survive’’ the truncation. This is done as follows.

Fix  $M_{14}$  and  $M_{15}$  to contain correct length padding for a message of length  $512 - 65 = 447$  bits. In this variant of the attack we hope that the ‘1’-bit padding is correct. The probability of this happening is  $1/2$ .

Also fix  $M_6 = M_7 = M_{10} = M_{11} = 0$  (these are set to zero for simplicity in this case; other values can be chosen). We shall control  $Q_6$ ,  $Q_7$ ,  $Q_{10}$ , and  $Q_{11}$  so that the same words of  $H^*$  are controlled. We set  $Q_6 = \alpha$ ,  $Q_7 = \beta$ ,  $Q_{10} = \gamma - \alpha \lll^{11}$ , and  $Q_{11} = \delta - \beta \lll^{12}$ , obtaining  $H_6^* = \alpha$ ,  $H_7^* = \beta$ ,  $H_{10}^* = \gamma$ , and  $H_{11}^* = \delta$ .

All message words  $M_i$  not mentioned are kept free. Compute  $W = \mathbf{B} \cdot M$ , and manipulate this vector (as above) to obtain free  $W_i$ ,  $6 \leq i < 16$ . The words  $W_i$ ,  $0 \leq i \leq 5$ , are fixed as

follows.

$$\begin{aligned}
W_0 &= -W_6 - 2W_7 - 2W_8 - W_9 + W_{12} - 2W_{14} - 2W_{15} + 2M_{14} + M_{15} \\
W_1 &= W_6 + W_8 - W_{10} + W_{13} + W_{14} + W_{15} - M_{14} - M_{15} \\
W_2 &= -W_7 - W_8 - W_{11} - W_{12} - W_{14} + 2M_{14} + M_{15} \\
W_3 &= -W_6 - 2W_7 - 2W_8 - W_9 + W_{12} - W_{13} - 2W_{14} - 2W_{15} + 2M_{14} + M_{15} \\
W_4 &= W_6 + W_7 + W_8 - W_{10} + W_{13} + W_{14} + W_{15} - 2M_{14} - M_{15} \\
W_5 &= -W_7 - W_{11} - W_{14} - M_{15} + 2M_{14}.
\end{aligned}$$

We may now select  $W_{15}, \dots, W_{12}$  arbitrarily and compute  $Q_{15}, \dots, Q_{12}$ . We then compute  $W_{11}$  and  $W_{10}$  so that  $Q_{11}$  and  $Q_{10}$  obtain the right values. We select  $W_9$  and  $W_8$  arbitrarily, and compute  $Q_9$  and  $Q_8$ . We then compute  $W_7$  and  $W_6$  so as to obtain the right values in  $Q_7$  and  $Q_6$ . Finally, we compute  $W_0, \dots, W_5$  according to the expressions above, and compute  $Q_5, \dots, Q_0$ .

We have now controlled four output words of the compression function, two of which are in the section that is not truncated away after the last message block has been processed. The message block has correct padding with probability  $1/2$ , and hence we have pseudo-attacks faster than expected, and faster than in the previous subsection. Pseudo-collisions can now be found in time  $2^{3n/8+1}$ , and pseudo-preimages (and -second preimages) can be found in time  $2^{3n/4+1}$ . This means that for  $n = 256$ , pseudo-collisions can be found in time about  $2^{97}$ , and pseudo-preimages can be found in time about  $2^{193}$ .

### 3.3 Further extensions

It seems difficult to extend the described technique to more than four output words of the compression function. There are still remaining degrees of freedom, but the inter-dependency between computations of words  $Q_i$  and words  $W_i$  seems to prevent further extensions. However, this is still work in progress.

It also seems difficult to extend the pseudo-attacks to real attacks on the hash function. This would require that the function  $f_0$  is taken into account, and in single-block attacks, 16 words of freedom are lost.

### 3.4 Examples

The following combination of chaining input ( $H$ ) and message block ( $M$ , includes padding, original length 447 bits) leads to a compression function output ( $H^*$ ) in BMW-256 with words 6, 7, 10, and 11 equal to zero.

$$\begin{aligned}
H &= (08bfd47a, f09ea8e7, 00f91374, abe0e8cf, 0e984813, 7bc3294e, e85cf22f, 9b171cb0, \\
&\quad 53b0ecd2, f67734fc, 9c130eaa, 8508814c, 08762814, 2739b073, 638d9137, 86b58c55)
\end{aligned}$$

$$\begin{aligned}
M &= (c2bad119, 8eb58e86, 70105431, fc599c67, 2b7e868d, 1f72d50c, 00000000, 00000000, \\
&\quad ebc36988, 115bf76c, 00000000, 00000000, 9872e2ed, a7d99cca, 000001bf, 00000000)
\end{aligned}$$

$$\begin{aligned}
H^* &= (523b3a78, 7b2f0e98, 61b77754, 9e93aa5d, c6fd1760, a194e500, 00000000, 00000000, \\
&\quad e6c075a0, 97818795, 00000000, 00000000, 59fcf808, 7f6f06eb, 99059c60, a653ec83)
\end{aligned}$$

Hence, the message digest if  $H$  were the initial value of BMW-256 would be

a075c0e695878197000000000000000008f8fc59eb066f7f609c059983ec53a6.

Finding this input pair took less than a second on a standard PC. In about  $2^{32.3}$  trials<sup>1</sup>, the following pair leading to a message digest with three words (12 bytes) equal to zero was found.

$H$  = (cb385300, eed84518, 2e720d56, 8210ffaa, ae14812e, bde0f95c, a899fe27, cae5ef44,  
4d714de1, 59f415c9, 0be8acd2, 91e02959, 98d8d24e, 42ff185b, 1aa19f6a, 023317bb)

$M$  = (4f178aee, e538d0bd, d676c083, 21739c56, 17851a4f, 2f0a1c70, 00000000, 00000000,  
c354456e, d90e91a9, 00000000, 00000000, 3c1aa40d, bd1ed3b9, 000001bf, 00000000)

The message digest is

d1acca6d000000000000000000000000072f5f8729c943d158126b1d7e51aaa1e.

## 4 Conclusion

The attacks described in this note are not a direct threat to the security of the Blue Midnight Wish hash function, since they require control over the chaining input to the compression function, or, put differently, over the initial value. However, we believe that the attacks show some undesirable properties in Blue Midnight Wish.

## References

- [1] R. P. Brent. An improved Monte Carlo factorization algorithm. *BIT*, 20(2):176–184, 1980.
- [2] R. W. Floyd. Nondeterministic Algorithms. *Journal of the Association for Computing Machinery*, 14(4):636–644, October 1967.
- [3] D. Gligoroski, V. Klima, S. J. Knapskog, M. El-Hadedy, J. Amundsen, and S. F. Mjølunes. Cryptographic Hash Function Blue Midnight Wish. SHA-3 Algorithm Submission, October 2008. Available: [http://people.item.ntnu.no/~daniilog/Hash/BMW/Supporting\\_Documentation/BlueMidnightWishDocumentation.pdf](http://people.item.ntnu.no/~daniilog/Hash/BMW/Supporting_Documentation/BlueMidnightWishDocumentation.pdf) (2009/04/07).
- [4] National Institute of Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. *Federal Register*, 27(212):62212–62220, November 2007. Available: [http://csrc.nist.gov/groups/ST/hash/documents/FR\\_Notice\\_Nov07.pdf](http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf) (2009/04/07).
- [5] G. Nivasch. Cycle detection using a stack. *Information Processing Letters*, 90(3):135–140, 2004.
- [6] S. S. Thomsen. A near-collision attack on the Blue Midnight Wish compression function, 2008. Available: <http://www.mat.dtu.dk/people/S.Thomsen/bmw/nc-compress.pdf> (2009/04/07).

---

<sup>1</sup>In fact, two searches were initiated; one succeeded, and one failed due to the padding not being correct. The total number of iterations needed was about  $2^{32.3}$