(Pseudo) Preimage Attack on Round-Reduced Grøstl Hash Function and Others (Extended Version)

Shuang Wu[1], Dengguo Feng[1], Wenling Wu[1], Jian Guo[2], Le Dong[1], and Jian Zou[1]

[1] State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences.
[2] Institute for Infocomm Research, Singapore.

wushuang@is.iscas.ac.cn

Abstract. The Grøstl hash function is one of the 5 final round candidates of the SHA-3 competition hosted by NIST. In this paper, we study the preimage resistance of the Grøstl hash function. We propose pseudo preimage attacks on Grøstl hash function for both 256-bit and 512-bit versions, *i.e.*, we need to choose the initial value in order to invert the hash function. Pseudo preimage attack on 5(out of 10)-round Grøstl-256 has a complexity of $(2^{244.85}, 2^{230.13})$ (in time and memory) and pseudo preimage attack on 8(out of 14)-round Grøstl-512 has a complexity of $(2^{507.32}, 2^{507.00})$. To the best of our knowledge, our attacks are the first (pseudo) preimage attacks on round-reduced Grøstl hash function, including its compression function and output transformation. These results are obtained by a variant of meet-in-the-middle preimage attack framework by Aoki and Sasaki. We also improve the time complexities of the preimage attacks against 5-round Whirlpool and 7-round AES hashes by Sasaki in FSE 2011.

Key words: hash function, meet-in-the-middle, preimage attack, Grøstl, Whirlpool, AES

1 Introduction

In FSE 2008, Gaëtan Leurent proposed the first preimage attack on the full MD4 hash function [12]. Based on this pioneering work, Aoki and Sasaki invented the technique of Meet-int-the-middle (MitM) preimage attack [2]. The basic idea of this technique is to divide the compression function into two concatenated sub-functions. The output values of two sub-functions can be independently calculated from the given input value in the forward direction and the backward direction. The steps of the forward and backward computation are called forward chunk and backward chunk. Then the MitM attack is applied to the output values of two sub-functions at the concatenating point of two chunks.

For hash functions based on block ciphers, the feedforward operations in the mode of operations like Davis-Meyer, Matyas-Meyer-Oseas and Miyaguchi-Preneel provide a chance for the applications of new technique called *splice-and-cut* [2]. The input and output of a compression function can be regarded as concatenated through the feed-forward operation in these modes of operations. Then the compression function is in the form of a circle and any step can be selected as either the starting point or the matching point.

Improvements have been developed on both the starting point and the matching point. The *initial structure* technique [16] (also called *message stealing* [7]) and the *local collision*[3] [15] technique allows two sub-functions to share several steps without violating the independency in computing their own values, which provides more attackable rounds. The *partial matching* technique [2, 16, 7, 1] takes advantage of the compression function's diffusion properties at the matching point. Due to slow diffusion of the Feistel-like round function, part of the state value can remain independent of the other chunk while proceeding with more reversed rounds. The

[3] The local collision technique was proposed by Joux et al. [5], which is originally used in the collision attacks. The similar idea can be used to construct the initial structure in the MitM preimage attack.

deterministic part of the state is used as the matching point. After finding a match of the partial values, the equality of the remaining part is calculated and checked. These techniques used in the MitM preimage attacks are illustrated in Fig. 1.

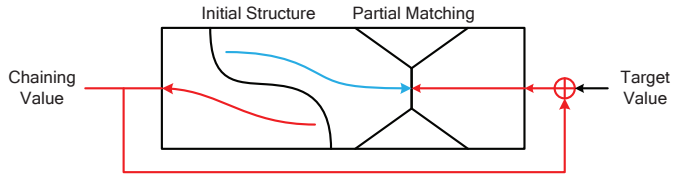

Fig. 1. Advanced techniques for MitM preimage attack

The MitM preimage attacks have been applied to full `HAVAL-3/4` [15], `MD4` [2, 7], `MD5` [16], `Tiger` [7], and round-reduced `HAS-160` [8], `RIPEMD` [21], `SHA-0/1` [3], `SHA-2` [7, 1]. The compression functions of these hash functions all use Feistel-like structures. In FSE 2011, Yu Sasaki proposed MitM preimage attack on `AES` hash mode for the first time [14]. He discussed how initial structure and partial matching can be used on `AES`-like structures and proposed direct applications to `AES` in different hash modes and round-reduced `Whirlpool` [4]. The development of the MitM attacks on hash functions has also inspired several attacks on block ciphers, such as `KTANTAN` [22] and `XTEA` [19].

Our contributions In this paper, we found a way to reduce the complexity of the MitM preimage attack on `AES`-like hash functions. By finding the optimal chunk separation with best balance between freedom degrees and the size of the matching point, the freedom degrees in the internal states are fully utilized.

`Grøstl` [6] is one of the five finalists in the third round of `SHA-3` [13] competition hosted by NIST. The `Grøstl` hash function has been tweaked in the third round. The original version is renamed to `Grøstl-0` and the tweaked version is called `Grøstl`.

We found that `Grøstl`'s round-reduced output transformation can be inverted using the MitM techniques. Then we noticed that if we can control the initial value, preimage of the output transformation can be connected with a compression function. The `Grøstl` hash function uses wide-pipe chaining values, so we can actually match 2n-bit chaining value with a time complexity less than 2^n compression function calls. Since the initial value is chosen by us, this attack is a pseudo preimage attack.

The matching of double-sized states are based on a method of variant generalized birthday attack. The special property of `Grøstl`'s compression function makes this approach possible. We found that the matching can be regarded as a special three-sum problem. Since the elements in one of the three sets can be restricted in a subspace, we can reduce the complexity to less than 2^n.

The comparison of previous best attacks and our attacks on `Grøstl` are shown in Table 1. Note that the attacks on `Grøstl-0` are not included in this table, since our attack is on the tweaked version.

We also improve the existing attacks against 5-round `Whirlpool` and 7-round `AES` hashing modes. While the previous result on 5-round `Whirlpool` applies to second preimage only, we improve the time complexity and also make the attack work for first preimages. We also improve the time complexity for the attacks against 7-round `AES` hashing modes. The details are presented in Appendix due to space limit.

Table 1. Comparison of the attacks on Grøstl-256 and Grøstl-512

Algorithm	Target	Attack Type	Rounds	Time	Memory	Source
Grøstl-256	Hash Function	Collision	3	2^{64}	-	[18]
	Compression Function	Semi-Free-Start Collision	6	2^{112}	2^{64}	[18]
	Permutation	Distinguisher	8	2^{48}	2^{8}	[17]
	Output Transformation	Preimage	5	2^{206}	2^{48}	Sect. 4.1
	Hash Function	Pseudo Preimage	5	$2^{244.85}$	$2^{230.13}$	Sect. 4
Grøstl-512	Hash Function	Collision	3	2^{192}	-	[18]
	Compression Function	Semi-Free-Start Collision	7	2^{152}	2^{56}	[17]
	Output Transformation	Preimage	8	2^{495}	2^{16}	Sect. 5.1
	Hash Function	Pseudo Preimage	8	$2^{507.32}$	$2^{507.00}$	Sect. 5

Outline of this paper In Sect. 2, we describe the specification of the Grøstl hash function. In Sect. 3, we introduce the attack outline of the pseudo preimage attack on reduced round Grøstl. Attacks on Grøstl-256 and Grøstl-512 are illustrated in Sect. 4 and Sect. 5 respectively. Sect. 6 is the conclusion.

2 Specification of Grøstl

Grøstl is a double-pipe design, i.e., the size of the chaining value ($2n$-bit) is twice as the hash size (n-bit). Message length should be less than $2^{73} - 577$ bits. The padding rule is not introduced here, since it's not important in our attack.

The compression function of Grøstl is written as:

$$F(H, M) = P(H \oplus M) \oplus Q(M) \oplus H$$

Where H is the chaining value and M is the message block, both are of $2n$ bits. After all message blocks are processed, the last chaining value X is used as input of the output transformation, which is written as

$$\Omega(X) = Trunc_n(P(X) \oplus X)$$

The right half of $P(X) \oplus X$ is used as the hash value. The compression function and output transformation are illustrated in Fig. 2.

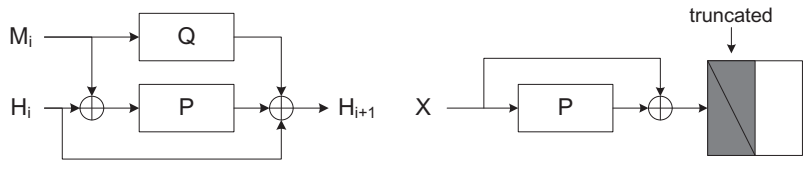

Fig. 2. Compression function and output transformation of Grøstl

P and Q are AES-like permutations with 8×8 and 8×16 sized state for Grøstl-256 and Grøstl-512 separately. Grøstl-256 uses 10-round P, Q and Grøstl-512 uses 14-round P, Q. The round function of the permutations consists of the four operations:

- SubBytes(SB): applies the Substitution-Box to each byte.
- ShiftBytes(SR): cyclically shifts the i-th row leftwards for i positions.
- MixBytes(MC): multiplies each column of the state matrix by an MDS matrix:

$$C = circ(02, 02, 03, 04, 05, 03, 05, 07)$$

- AddRoundConstant(AC): XOR the round constant to the state.

The shift vectors used in P and Q are different. P in Grøstl-256 uses (0,1,2,3,4,5,6,7) and P in Grøstl-512 uses (0,1,2,3,4,5,6,11). In the description of our attack, we skip Q's detail since it's not required.

An important property of the compression function has been pointed out in the submission document of Grøstl hash function [6]. Note that with $H' = H \oplus M$, the compression function can be written as

$$F(H, M) = P(H') \oplus H' \oplus Q(M) \oplus M.$$

So the generic preimage attack on the compression function with $2n$-bit state costs 2^n computations, since solving the equation $F(H, M) = T$ can be regarded as a birthday problem. Then the collision attack on the compression function costs $2^{2n/3}$ computations, since $F(H_1, M_1) \oplus F(H_2, M_2) = 0$ is a (four-sum) generalized birthday problem [20].

3 Outline of the Attack on the Grøstl Hash Function

Suppose the hash size is n-bit and the state size is $2n$-bit. In order to find a pseudo preimage (H, M) of the Grøstl hash function, let $X = F(H, M)$, then X is the preimage of the output transformation: $P(X) \oplus X = *||T$ where T is the target hash value and $*$ stands for arbitrary n-bit value. With $H' = H \oplus M$, we have

$$(P(H') \oplus H') \oplus (Q(M) \oplus M) \oplus X = 0 \tag{1}$$

If we have collected enough candidates for $P(H') \oplus H'$, $Q(M) \oplus M$ and X, the pseudo preimage attack turns into a three-sum problem. As we know, there is no generic solution for three-sum problem faster than birthday attack. But if we can restrict $P(H') \oplus H'$ in a subspace, it is possible to break the birthday bound. Here we restrict $P(H') \oplus H'$ in a subspace by finding its partial zero preimages.

As illustrated in Fig. 3, the attack process is similar to the generalized birthday attack [20]. With four parameters x_1, x_2, x_3 and b, this attack can be described in four steps:

1. Find 2^{x_1} preimages X of the output transformation and store them in lookup table L_1.
2. Find 2^{x_3} H' such that leftmost b bits of $P(H') \oplus H'$ are all zero. Then store all $P(H') \oplus H'$ and H' in lookup table L_2. This step can be regarded as finding partial zero preimages on $P(H') \oplus H'$.
3. Choose 2^{x_2} random M with correct padding and calculate $Q(M) \oplus M$. Then check if there is an X in L_1 with the same leftmost b bits as $Q(M) \oplus M$. We expect to find $2^{x_1+x_2-b}$ partial matches $Q(M) \oplus M \oplus X$ here, whose left most b bits are all zero.
4. For each of the $2^{x_1+x_2-b}$ $Q(M) \oplus M \oplus X$ found in step 3, check if its remaining $(2n - b)$-bit value can be found in L_2.

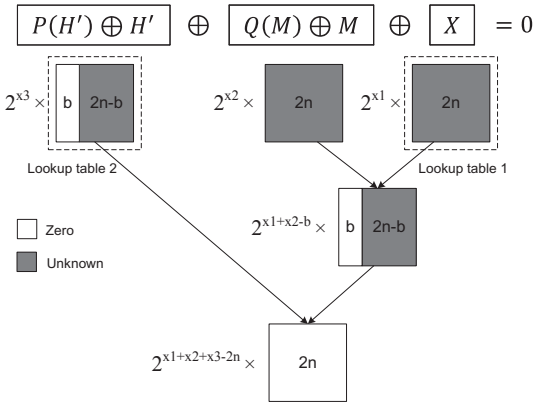

Fig. 3. Outline for pseudo preimage attack on the `Grøstl` hash function

Once a final match is found, we have H', M and X which satisfies equation (1). So, $(H' \oplus M, M)$ is a pseudo preimage of `Grøstl`.

Note that to find an X is to find an n-bit partial preimage of $P(X) \oplus X$ and the truncation bits are fixed (the leftmost n-bits are truncated). But for $P(H') \oplus H'$, it's not necessary to find partial preimage for the leftmost b bits. In fact, we can choose any b bits as the zero bits. We will further discuss the differences between fixed position and chosen position partial preimage attacks later.

Suppose that for `Grøstl` with $2n$-bit state, it takes $2^{C_1(2n,n)}$ computations to find a fixed position n-bit partial preimage and it takes $2^{C_2(2n,b)}$ computations to find a chosen position b-bit partial preimage of $P(X) \oplus X$. Now we calculate the complexity for each of the four attacking steps:

1. Step 1, building the look-up table 1 takes $2^{x_1+C_1(2n,n)}$ computations and 2^{x_1} memory.
2. Step 2, building the look-up table 2 takes $2^{x_3+C_2(2n,b)}$ computations and 2^{x_3} memory.
3. Step 3, calculating $Q(M) \oplus M$ for 2^{x_2} M and checking the partial match in table 1 takes 2^{x_2} Q calls, which is equivalent to 2^{x_2-1} compression function calls.
4. Step 4, checking the final match for $2^{x_1+x_2-b}$ candidates requires $2^{x_1+x_2-b}$ table look-ups, which can be equivalently regarded as $2^{x_1+x_2-b}C_{TL}$ compression function calls. C_{TL} is the complexity of one table lookup, where unit one is one compression function call. For 5-round `Grøstl-256` and 8-round `Grøstl-512`(the attacked versions), C_{TL} is chosen as 1/640 and 1/2048 respectively[4].

Then the overall complexity is:

$$2^{x_1+C_1(2n,n)} + 2^{x_3+C_2(2n,b)} + 2^{x_2-1} + 2^{x_1+x_2-b} \cdot C_{TL} \tag{2}$$

with memory requirement of $2^{x_1} + 2^{x_3}$.

In the following sections, we first show how to find partial preimages of the function $P(X) \oplus X$ and calculate the complexity $C_1(2n, n)$ and $C_2(2n, b)$. Then we need to choose optimal parameters x_1, x_2, x_3 and b to minimize the complexity with the restriction of $x_1 + x_2 + x_3 \geq 2n$ and $0 \leq b \leq 2n$. Since in order to find one final match, we need $2^{x_1+x_2+x_3-2n} \geq 1 \Rightarrow x_1+x_2+x_3 \geq 2n$.

[4] The constant C_{TL} is chosen as the upper bound of the complexity that one table lookup takes, due to the fact that 5-round `Grøstl-256` software implementation composes of $(8 * 8) * 5 * 2 = 640$ s-box lookups, and other operations. In 8-round Grostl-512, there are $(8 * 16) * 8 * 2 = 2048$ s-box lookups.

4 Pseudo Preimage Attack on 5-round Grøstl-256

In this section, first, we introduce the preimage attack on the output transformation, *i.e.*, the fixed position partial preimage attack on $P(X) \oplus X$ and calculate the complexity $C_1(512, 256)$. Then we introduce the chosen position partial preimage attack on $P(H') \oplus H'$ and give the expression of the function $f(b) = C_2(512, b)$. At last, we try to minimize the overall complexity by finding proper parameters for the generic attack introduced in Section 3.

4.1 Fixed Position Partial Preimage Attack on $P(X) \oplus X$

The chunk separation for this attack is shown in Fig. 4. Note that the yellow cells with a diagonal line are the truncated bytes, which can be regarded as free variables. In the last state of Fig. 4, the equations for the truncated byte can be directly removed since they are automatically fulfilled. The size of the full match is 256-bits for this MitM attack.

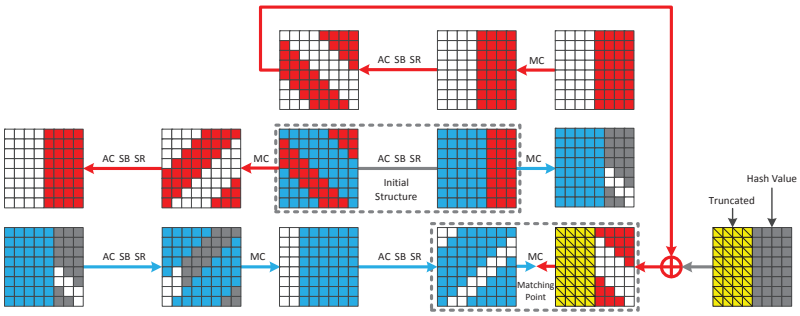

Fig. 4. Chunk separation of preimage attack on Grøstl-256's output transformation

The Colors in the Chunk Separation First, we explain what the colors stand for. Actually, we use the same colors as in [14] to illustrate the chunk separations. The blue bytes in the forward chunk can be determined by the blue bytes in the initial structure. The white color in the forward chunk stands for the bytes whose values are affected by both red bytes and blue bytes in the initial structure, and can't be pre-computed until the partial match is found. Similarly, in the backward chunk, red and white cells stand for the certain and uncertain bytes. The gray cells are constant bytes in the target value, the chaining value and the initial structure, which are known or can be chosen before the MitM attack.

Freedom Degrees and Size of the Matching Point Before we apply the MitM attack, we need to know the freedom degrees in the forward and backward directions and the bit size of the matching point. The calculation method has been explained in [14]. More details about this is in Appendix A.

With the method introduced in appendix A, we can find that , in Fig. 4, there are $D_2 = 2^{48}$ and $D_1 = 2^{64}$ freedom degrees in red and blue bytes respectively. In each of the four available columns, there are two bytes of matching point. So the size of the matching point is $m = 4 \times (2 \times 8) = 64$ bits.

The Attack Algorithm and Its Complexity In this section, we consider a generic MitM attack algorithm with partial matching technique. Suppose there are 2^{D_1} and 2^{D_2} freedom

degrees in the forward and backward chunks. The size of the matching point is m-bit and the full matching size is b-bit. Without loss of generality, assume that $D_1 \geq D_2$. Note that if $D_1 + D_2 \geq b$, we can't fully use all the freedom degrees. Here we use d_1 and d_2 to denote the actually used freedom degrees:

$$(d_1, d_2) = \begin{cases} (D_1, D_2), & if D_1 + D_2 \leq b; \\ (b/2, b/2), & if D_1 + D_2 > b \ \ and \ \ D_2 \geq b/2; \\ (b - D_2, D_2), & if D_1 + D_2 > b \ \ and \ \ D_2 < b/2. \end{cases} \tag{3}$$

This MitM preimage attack can be described in four steps.

1. Choose random constants in the initial structure.
2. With the chosen constants, for all 2^{d_2} values v_j^2 of the forward direction, calculate all the partial values p_j^2 and the full values f_j^2 at the matching point and store all the pairs (v_j^2, p_j^2) in a look up table L;
3. For all 2^{d_1} values v_i^1 of the backward direction, calculate p_i^1. Then check if p_i^1 is in table L. If we found one partial match that $p_i^1 = p_j^2$ for some j, calculate the full value f_i^1 using v_i^1 and check if $f_i^1 = f_j^2$;
4. If no full match has been found yet, go to step 1.

Then we calculate the complexity. Step 2 costs 2^{d_2} f_2 calls and 2^{d_2} memory. Step 3 costs 2^{d_1} f_1 calls. Consider two kinds of circumstances separately.

- If $d_1 + d_2 \geq m$. After step 3 is done, we expect $2^{d_1+d_2-m}$ good candidates that satisfy the m-bit matching point. Now check if the full value of all good candidates are matched. This step requires $2^{d_1+d_2-m}$ computations. The probability that a good candidate is a full match is 2^{m-b}. Then the probability that there exists one full match in $2^{d_1+d_2-m}$ good candidates is about $2^{(d_1+d_2-m)+(m-b)} = 2^{d_1+d_2-b}$. So, we need to repeat the attack $2^{b-d_1-d_2}$ times in order to find a full match. The complexity is:

$$2^{b-d_1-d_2} \cdot (2^{d_1} + 2^{d_2} + 2^{d_1+d_2-m}) = 2^b \cdot (2^{-d_1} + 2^{-d_2} + 2^{-m})$$

- If $d_1 + d_2 < m$. After step 3 is done, we can find one good candidate with probability of $2^{d_1+d_2-m}$. So, we need to repeat the attack $2^{m-d_1-d_2}$ times to find one good candidate, then we calculate the full value of the good candidate at the matching point to check if it is a full match, which cost one computation. So the complexity to find one good candidate and check its full value is $2^{m-d_1-d_2}(2^{d_1} + 2^{d_2}) + 1$. Then find and check 2^{b-m} good candidates to get a full match. The complexity is:

$$2^{b-m} \cdot (2^{m-d_1-d_2}(2^{d_1} + 2^{d_2}) + 1) = 2^b \cdot (2^{-d_1} + 2^{-d_2} + 2^{-m})$$

So, no matter in which case, the complexity to find one full match using this algorithm is always

$$2^b \cdot (2^{-d_1} + 2^{-d_2} + 2^{-m}) \tag{4}$$

computations and 2^{d_2} memory.

Application to `Grøstl`'s Output Transformation In Fig. 4, the freedom degrees are $D_1 = 48, D_2 = 64$, the partial and full matching size are $m = 64$ and $b = 256$ bits. Using the attack algorithm introduced in Section 4.1, we can calculate the complexity to invert 5-round `Grøstl`'s output transformation. Here the complexity is measured by compression function calls. In the MitM attack it takes about half P calls, `i.e.` 1/4 compression function calls to evaluate the matching point for one direction. Thus we can multiply 2^{-2} to the complexity: $2^{C_1(512,256)} = 2^{-2} \cdot 2^{256}(2^{-64} + 2^{-48} + 2^{-64}) \approx 2^{206}$ compression function calls with 2^{48} memory.

On the Choice of the Chunk Separation We can prove that our chunk separation in Fig. 4 is optimal, which minimizes the complexity of inverting the output transformation.

Suppose there are b blue bytes and r red bytes in each column of the matching point. Then we show the relation between b, r, freedom degrees D_1, D_2 and the partial matching size m.

In the forward direction, r red bytes in one column of the matching point $\xrightarrow{AC,SB,SR,MC}$ r full red columns $\xrightarrow{AC,SB,SR}$ r red bytes in one column. Here we stops at the left end of the initial structure. In order to produce at least one byte of freedom degrees in the blue color, there are at least $r+1$ blue columns in the initial structure. Then there would be at most $8-(r+1) = 7-r$ red columns in the initial structure.

In the backward direction, b blue bytes in one column of the matching point $\xrightarrow{SR^{-1},SB^{-1},AC^{-1}}$ $8-b$ white columns $\xrightarrow{MC^{-1},SR^{-1},SB^{-1},AC^{-1}}$ $8-b$ white bytes in each columns.

Now we count the freedom degrees. There are $(7-r)$ red columns in the initial structure and each column produces $8-b$ free bytes. So, freedom degrees in red color is $D_2 = 8(7-r)(8-b)$ bits. The minimum freedom degrees in the blue color here is $D_1 = 2^{64}$. Size of the matching point in one column is $8(b+r-8)$ bits, so there are $4 \times 8(b+r-8)$ bits of matching point in total.

So the complexity is $2^{-2} \cdot 2^{256}(2^{-64} + 2^{-8(7-r)(8-b)} + 2^{-32(b+r-8)})$. The minimum complexity is 2^{206} when $b = 6, r = 4$ or $b = 5, r = 5$. Fig. 4 is the case of $b = 6, r = 4$.

4.2 Chosen Position Partial Preimage Attack on $P(H') \oplus H'$

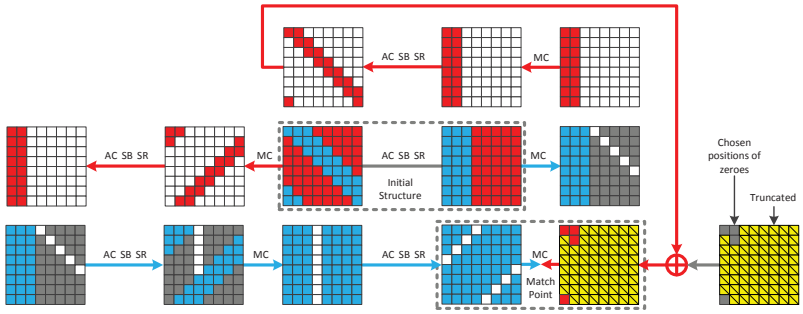

Fig. 5. Chunk separation of chosen position partial preimage attack on $P(H') \oplus H'$ for `Grøstl-256`

Now, consider the attack model of chosen position partial preimage. In the partial preimage attack of $P(H') \oplus H'$, we can choose the positions of the target bits. In order to minimize the complexity, we choose this chunk separation to maximize the size of the matching point $m(b)$ within all possible b target bits.

First, we discuss the size of matching point and chosen positions in one column. If less than 8 bits of the red byte in one column are chosen, no matching point can be derived. if $b > 8$ bits of the red bytes are chosen, as explained in appendix A, there are $b-8$ bits of matching point. Since there are only two red bytes in one column in the last state of Fig. 5, even if $b > 16$, no more than 8 bits of matching point can be derived. In order to maximize $m(b)$, we choose at most 2 red bytes in one column and then chose the red bytes from another column. When $b > 128$, $m(b) = 64$, because there are 64 bits of matching point in total. The graph of $m(b)$ is shown in Fig. 6.

In this Figure, freedom degrees in the red and blue color are $D_2 = 40$ and $D_1 = 64$. Then we can calculate the complexity of chosen position partial preimage:

$$2^{C_2(512,b)} = 2^{-2} \cdot 2^b (2^{-d_1} + 2^{-d_2} + 2^{-m(b)})$$

where d_1 and d_2 are chosen according to equation (3), `i.e.`:

$$(d_1, d_2) = \begin{cases} (64, 40), & if \quad b \geq 104; \\ (b - 40, 40), & if \quad 80 \leq b < 104; \\ (b/2, b/2), & if \quad b < 80. \end{cases}$$

The graph of $C_2(512, b)$ is shown in Fig. 7. When $b > 80$, $C_2(512, b) \approx b - 42$.

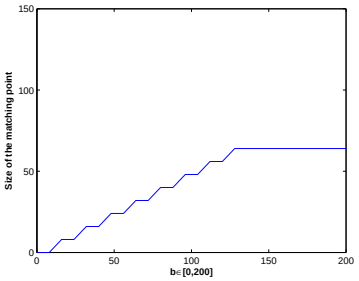

Fig. 6. Size of the matching point for chosen position truncations for `Grøstl-256`

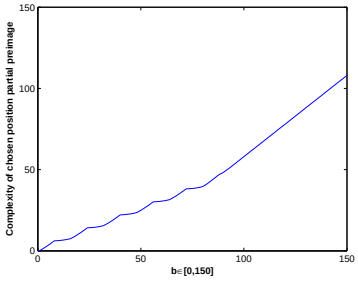

Fig. 7. Complexity of chosen position partial preimage of $P(H') \oplus H'$ for `Grøstl-256`

4.3 Minimizing the Overall Complexity

By now, we have found $C_1(512, 256)$ and $C_2(512, b)$. So we can start to deal with the overall complexity in equation (2). In the expression of the complexity, b can be integers from 0 to 512. For all $b \in [0, 512]$, optimal x_1, x_2 and x_3 are chosen to minimize the overall complexity. The graph of the minimum overall complexity for $b \in [0, 120]$ is shown in Fig. 8.

When $b = 31, x_1 \approx 36.93, x_2 \approx 244.93$ and $x_3 \approx 230.13$, the complexity is the lowest: $2^{244.85}$ compression function calls. Memory requirement is $2^{230.13}$. The chosen positions for the 31 bits ≈ 4 bytes are marked in Fig. 5.

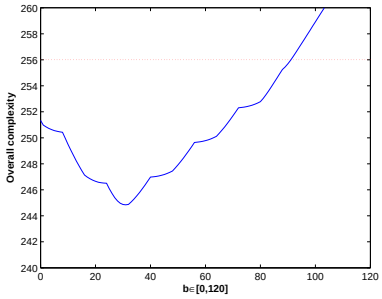

Fig. 8. Overall complexity of pseudo preimage attack on 5-round `Grøstl-256`

5 Pseudo Preimage Attack on 8-round Grøstl-512

The attack on Grøstl-512 uses the same method for the three-sum phase as in the attack on Grøstl-256. Here we skip the details of the attack algorithm and introduce the difference between the attacks on them only.

5.1 Fixed Position Partial Preimage Attack on $P(X) \oplus X$

The chunk separation for 8-round Grøstl-512 is shown in Fig. 9. Note that in this figure, we use a 2-round initial structure. Freedom degrees in the red and blue bytes are both 2^{16}. There are 4 bytes of matching point in total, as shown in Table 2 in Appendix B.

The parameters for the MitM preimage attack on the output transformation are $D_1 = D_2 = 16, m = 32$ and $n = b = 512$. So the complexity is $2^{C_1(1024,512)} = 2^{-2} \cdot 2^{512}(2^{-16} + 2^{-16} + 2^{-32}) \approx 2^{495}$ compression function calls and 2^{16} memory.

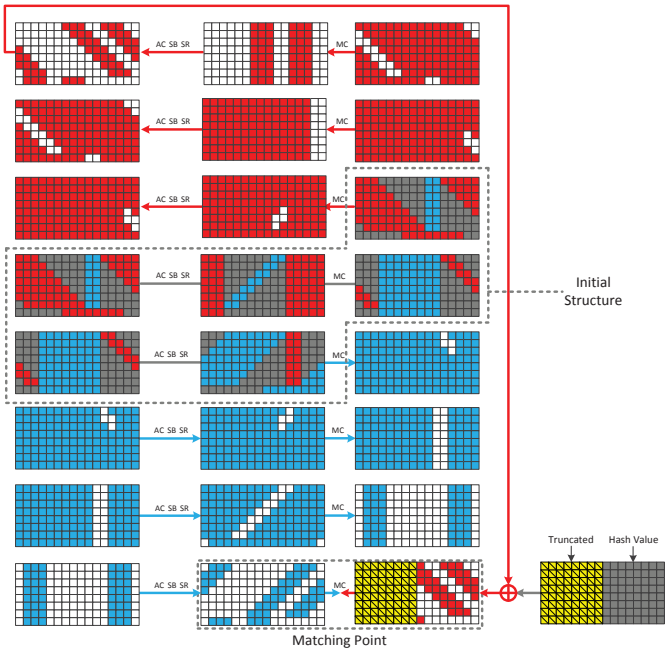

Fig. 9. Chunk separation of preimage attack on Grøstl-512's output transformation

On the Choice of the Chunk Separation Actually, we searched for all the possible patterns of the chunk separation for 8-round Grøstl-512. The chunk separation in Fig. 9 is one of the best we found. The search algorithm is as follows:

Step 1. Search for the matching point.

We want to find good candidates in all the possible positions of the white columns in round 2 and round 6. Since there are 32 columns in two states, there are 2^{32} patterns in total.

For each of the pattern of white columns, we can calculate round 2 backward and round 6 forward and check if there are at least two byte of matching point. After the search for all the 2^{32} patterns, we found 1322 patterns with at least two bytes of matching point.

Step 2. Search for the initial structure.

Considering the mirror image and rotational similarity, there are only 120 distinct patterns in all the 1322 patterns of matching point. For each of the 120 patterns, we calculate forward from round 2 and backward from round 6.

If there is one white column in round 2, the number of possible patterns of the white bytes in the same column of round 3 is $2^8 - 1$, since there must be at least one white byte in this column. So size of the search space is $(2^8 - 1)^w$, where w is the number of white columns in both round 2 and round 6. In the 120 possible patterns, w is no more than 4, so the search space is at most $2^{32} \cdot 120 \approx 2^{39}$.

Using early-abort trick, we can directly skip some bad patterns in round 2 without knowing the pattern in round 6. Then the search space is reduced again and the search is practical.

5.2 Chosen Position Partial Preimage Attack on $P(H') \oplus H'$

For chosen position partial preimage, we use another chunk separation in Fig. 10.

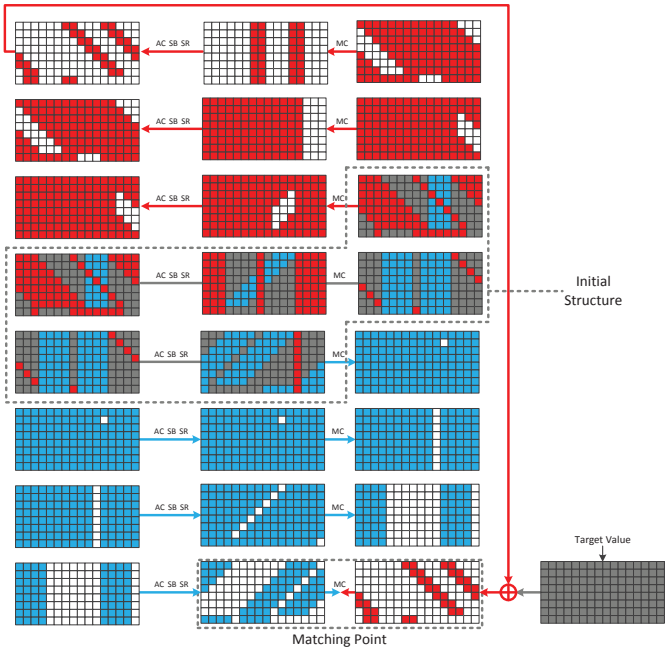

Fig. 10. Chunk separation of chosen position partial preimage attack on $P(H') \oplus H'$ for `Grøstl-512`

The freedom degrees for the MitM preimage attack are $D_1 = 24, D_2 = 8$. The distribution of the matching point bytes are shown in Table 3 and The graph of $m(b)$ is in Fig. 11. Then we can calculate the complexity of chosen position partial preimage:

$$2^{C_2(1024,b)} = 2^{-2} \cdot 2^b (2^{-d_1} + 2^{-d_2} + 2^{-m(b)})$$

where d_1 and d_2 are chosen according to equation (3), *i.e.*,

$$(d_1, d_2) = \begin{cases} (24, 8), & if \quad b \geq 32; \\ (b-8, 8), & if \quad 16 \leq b < 32; \\ (b/2, b/2), & if \quad b < 16. \end{cases}$$

The graphs for $m(b)$ and $C_2(1024, b)$ are in Fig. 11 and Fig. 12. The figures and tables are in Appendix B.

5.3 Minimizing the Overall Complexity

With the value and expression of $C_1(1024, 512)$ and $C_2(1024, b)$, we can deal with the overall complexity like we have done for `Grøstl-256`. The minimum overall complexity for different b is shown in Fig. 13.

When $b = 0, x_1 \approx 10.50, x_2 \approx 506.50$ and $x_3 \approx 507.00$, the overall complexity is the lowest: $2^{507.32}$. Memory requirement is $2^{507.00}$.

6 Conclusion

In this paper, we proposed pseudo preimage attacks on the hash functions of 5-round `Grøstl-256` and 8-round `Grøstl-512`. This is the first pseudo preimage attack on round-reduced `Grøstl` hash function, which is a wide-pipe design.

In order to invert the wide-pipe hash function, we have to match $2n$-bit state value with less than 2^n computations. This is achieved by exploiting the special property of the `Grøstl` compression function. After collecting enough partial preimages on the component $P(X) \oplus X$, the double-sized state values are matched using a variant of the generalized birthday attack.

There is an interesting observation that this attack works with any function Q. Thus our attack can be applied to the `Grøstl` hash function with round-reduced permutation P and full-round permutation Q. However, our attacks do not threat any security claims of `Grøstl`.

Acknowledgement The authors would like to thank Kazumaro Aoki, Keting Jia, Mohammad Ali Orumiehchiha, Somitra Sanadhya, and Chunhua Su for their inspiring suggestions during the ASK 2011 workshop. The authors would also thank Lei Wang for useful discussions.

References

1. Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for Step-Reduced SHA-2. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 578–597. Springer, 2009.
2. Kazumaro Aoki and Yu Sasaki. Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *LNCS*, pages 103–119. Springer, 2008.
3. Kazumaro Aoki and Yu Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 70–89. Springer, 2009.
4. Paulo S.L.M. Barreto and Vincent Rijmen. The whirlpool hashing function. Submission to NESSIE, September 2000.
5. Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *LNCS*, pages 56–71. Springer, 1998.
6. Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl – a SHA-3 candidate. Submission to NIST, 2008.
7. Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 56–75. Springer, 2010.
8. Deukjo Hong, Bonwook Koo, and Yu Sasaki. Improved Preimage Attack for 68-Step HAS-160. In Donghoon Lee and Seokhie Hong, editors, *ICISC*, volume 5984 of *LNCS*, pages 332–348. Springer, 2009.
9. Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *LNCS*, pages 306–316. Springer, 2004.
10. John Kelsey and Bruce Schneier. Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 474–490. Springer, 2005.
11. Aggelos Kiayias, editor. *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558 of *LNCS*. Springer, 2011.
12. Gaëtan Leurent. MD4 is Not One-Way. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *LNCS*, pages 412–428. Springer, 2008.

13. 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, Nov. 2007. Available: `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf` (2008/10/17).
14. Yu Sasaki. Meet-in-the-Middle Preimage Attacks on AES Hashing Modes and an Application to Whirlpool. In Antoine Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 378–396. Springer, 2011.
15. Yu Sasaki and Kazumaro Aoki. Preimage Attacks on 3, 4, and 5-Pass HAVAL. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *LNCS*, pages 253–271. Springer, 2008.
16. Yu Sasaki and Kazumaro Aoki. Finding Preimages in Full MD5 Faster Than Exhaustive Search. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *LNCS*, pages 134–152. Springer, 2009.
17. Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. New Non-Ideal Properties of AES-Based Permutations: Applications to ECHO and Grøstl. In *ASIACRYPT*, volume 6477 of *LNCS*, pages 38–55. Springer, 2010.
18. Martin Schläffer. Updated Differential Analysis of Grøstl. Grøstl website, January 2011.
19. Gautham Sekar, Nicky Mouha, Vesselin Velichkov, and Bart Preneel. Meet-in-the-Middle Attacks on Reduced-Round XTEA. In Kiayias [11], pages 250–267.
20. David Wagner. A Generalized Birthday Problem. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.
21. Lei Wang, Yu Sasaki, Wataru Komatsubara, Kazuo Ohta, and Kazuo Sakiyama. (Second) Preimage Attacks on Step-Reduced RIPEMD/RIPEMD-128 with a New Local-Collision Approach. In Kiayias [11], pages 197–212.
22. Lei Wei, Christian Rechberger, Jian Guo, Hongjun Wu, Huaxiong Wang, and San Ling. Improved Meet-in-the-Middle Cryptanalysis of KTANTAN (Poster). In Udaya Parampalli and Philip Hawkes, editors, *ACISP*, volume 6812 of *LNCS*, pages 433–438. Springer, 2011.

A Calculation of Freedom Degrees and Size of the Matching Point

Calculating Freedom Degrees Compared to original attack, we have less constants in the initial structure. In Fig. 4, there is no constant in the initial structure. Before the MC operation that produces uncertain (white) bytes in the forward chunk, there are 24 red bytes. In order to maintain 18 constant (gray) bytes after the MC operation, it is equivalent to solve such a equation group with 24 variables and 18 equations:

$$C \cdot \begin{pmatrix} r_0 & r_8 & r_{16} \\ r_1 & r_9 & r_{17} \\ r_2 & r_{10} & r_{18} \\ r_3 & r_{11} & r_{19} \\ r_4 & r_{12} & r_{20} \\ r_5 & r_{13} & r_{21} \\ r_6 & r_{14} & r_{22} \\ r_7 & r_{15} & r_{23} \end{pmatrix} = \begin{pmatrix} c_0 & c_8 & c_{16} \\ c_1 & c_9 & c_{17} \\ c_2 & c_{10} & c_{18} \\ c_3 & c_{11} & c_{19} \\ * & c_{12} & c_{20} \\ * & * & c_{21} \\ c_6 & * & * \\ c_7 & c_{15} & * \end{pmatrix} \tag{5}$$

where C is the MDS matrix, $\{r_i\}$ are the values of red bytes, $\{c_i\}$ are constants and $*$ are arbitrary values we don't care. This equation group has $2^{8\cdot(24-18)} = 2^{48}$ solutions, which are the freedom degrees in the red bytes, `i.e.` the backward chunk. Similarly, by observing 40 variables and 32 equations, the freedom degrees in the blue bytes can be calculated as $2^{8\cdot(40-32)} = 2^{64}$.

Calculating Size of the Matching Point First, we need to explain how the partial matching through MR operation works. Use the first column of the matching point in Fig. 4 as an example, our target is to find proper values that satisfies the following equation:

$$C \cdot \left(x_0\ x_1\ x_2\ x_3\ \underline{x_4}\ \underline{x_5}\ x_6\ x_7 \right)^T = \left(y_0\ \underline{y_1}\ \underline{y_2}\ \underline{y_3}\ \underline{y_4}\ y_5\ y_6\ y_7 \right)^T \tag{6}$$

where $x_0, x_1, x_2, x_3, x_6, x_7, y_0, y_5, y_6, y_7$ are known bytes and $\underline{x_4}, \underline{x_5}, \underline{y_1}, \underline{y_2}, \underline{y_3}, \underline{y_4}$ are uncertain bytes. So only four equations of y_0, y_5, y_6 and y_7 are useful to us:

$$05\underline{x_4} + 03\underline{x_5} = 02x_0 + 02x_1 + 03x_2 + 04x_3 + 05x_6 + 07x_7 + y_0 \quad (7)$$
$$07\underline{x_4} + 02\underline{x_5} = 04x_0 + 05x_1 + 03x_2 + 05x_3 + 02x_6 + 03x_7 + y_5 \quad (8)$$
$$05\underline{x_4} + 07\underline{x_5} = 03x_0 + 04x_1 + 05x_2 + 03x_3 + 02x_6 + 02x_7 + y_6 \quad (9)$$
$$03\underline{x_4} + 05\underline{x_5} = 02x_0 + 03x_1 + 04x_2 + 05x_3 + 07x_6 + 02x_7 + y_7 \quad (10)$$

From equations (7)(8), obtain $\underline{x_4}, \underline{x_5}$ with linear combinations of the known bytes:

$$\underline{x_4} = F7x_0 + A5x_1 + 00x_2 + 52x_3 + A5x_6 + 53x_7 + 52y_0 + 52y_5$$
$$\underline{x_5} = F6x_0 + A4x_1 + 8Cx_2 + 50x_3 + 2Ax_6 + DDx_7 + DFy_0 + 52y_5$$

Then equation (10) can be rewritten as:

$$F1x_0 + 52x_1 + 8Cx_2 + A9x_3 + D3x_6 + 23x_7 = 2Ay_0 + A4y_5 + y_6 \quad (11)$$
$$03x_0 + F5x_1 + 8Ex_2 + F8x_3 + 71x_6 + 73x_7 = 78y_0 + F7y_5 + y_7 \quad (12)$$

Equations (11) and (12) are used as the matching point, since they provide equations of the known bytes that can be pre-computed, stored separately and then checked using table look-ups. Here the matching point is not directly truncated from the state value. In Fig. 4, two bytes of matching point can be derived from one column. At most 8 bytes (64 bits) of matching point can be found, since there are only four available columns.

Suppose there are b and r known bytes in one column of the input and output values of the MC at the matching point. $8 - b$ uncertain bytes are regarded as variables and r known red bytes can provide r equations. $8 - b$ of the equations are used to determine values of the variables. Then the remaining $r - (8 - b) = r + b - 8$ equations are on the known bytes, which are regarded as the matching point. Note that if $b + r \leq 8$, no matching point can be derived from this column. Otherwise, there are $b + r - 8$ bytes of matching point in this column.

If size of the known bits b' and r' are not exact multiples of 8, we can further split the linear equations on bytes into linear equations on bits. The bit size of matching point can be calculated as $b' + r' - 64$.

B Figures and Tables for `Grøstl-512`

Table 2. The matching point in Fig. 9

column index	9	10	11	12	13	14	15	16
blue bytes	4	3	3	4	5	5	4	3
red bytes	4	3	3	4	5	5	4	3
sum	8	6	6	8	10	10	8	6
matching point(bytes)	0	0	0	0	2	2	0	0

C Preimage Attack on round-reduced `Whirlpool`

C.1 Specification of `Whirlpool`

`Whirlpool` uses MD-strengthening structure, with narrow-pipe chaining value and no block counters. So it is vulnerable to generic attack, like the expandable messages [10] and multi-target pseudo preimage [12] attack. We will talk about the details later.

Table 3. matching points in Fig. 10

column index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
blue bytes	5	4	3	2	0	2	3	4	5	4	5	6	6	6	5	4
red bytes	2	3	2	0	0	0	2	3	2	2	2	3	4	3	2	2
sum	7	7	5	2	0	2	5	7	7	6	7	9	10	9	7	6
matching point(bytes)	0	0	0	0	0	0	0	0	0	0	0	1	2	1	0	0

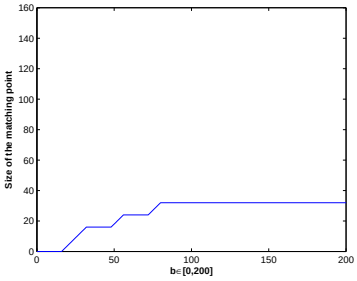

Fig. 11. Size of the matching point for chosen position truncations for `Grøstl-512`

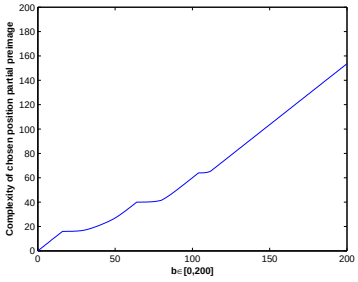

Fig. 12. Complexity of chosen position partial preimage of $P(H') \oplus H'$ for `Grøstl-512`

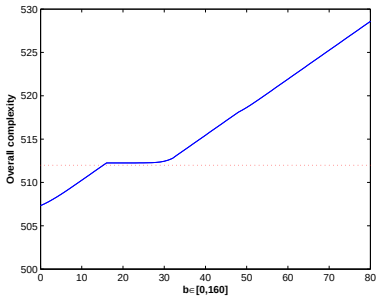

Fig. 13. Overall complexity of pseudo preimage attack on 8-round `Grøstl-512`

`Whirlpool` accepts any message with less than 2^{256} bits as input and the 256-bit binary expression of bit length is padded according to MD-strengthening, `i.e.` $M||1||0^*||length$. Size of the message block, the chaining value and the hash value is 512-bit.

Compression function of `Whirlpool` can be regarded as a block cipher called W in Miyaguchi-Preneel mode.

$$F(H, M) = W_H(M) \oplus M \oplus H$$

where block cipher W use `AES`-like iteration with 8×8 state of bytes and the $(8i + j)$-th input byte of the message block is placed at the i-th row and j-th column of the state. Each round consists of four operations:

- SubBytes(SB): applies the Substitution-Box to each byte.
- ShiftColumns(SC): cyclically shift the i-th column downwards for i positions.
- MixRows(MR): multiply each row of the state matrix by an MDS matrix

$$C = circ(01, 01, 04, 01, 08, 05, 02, 09)$$

- AddRoundKey(AK): XOR the round key to the state.

Since the key schedule is not important in our attack, the description is omitted.

C.2 Improved Second Preimage Attack on `Whirlpool`

In [14], Yu Sasaki proposed a second preimage attack on 5-round `Whirlpool` using the MitM approach. In their attack, there are only 2^8 freedom degrees in both chunks, but the size of matching point is much larger (40 bytes=320 bits). The comparison of the preimage attacks on `Whirlpool` is shown in Table 4.

Table 4. Comparison of the preimage attacks on `Whirlpool`

Attack Type	Rounds	Time	Memory	Source
Second Preimage	5	2^{504}	2^8	[14]
Second Preimage	5	2^{448}	2^{64}	this section
Preimage	5	$2^{481.5}$	2^{64}	this section

In this section, we propose an improved chunk separation with more freedom degrees and a smaller matching point in Fig. 14.

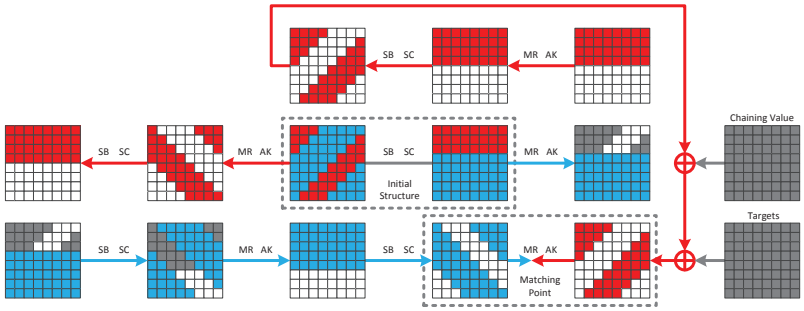

Fig. 14. Chunk separation for improved 2nd-preimage attack on 5-round `Whirlpool`

We use the same colors as in [14] to illustrate the chunk separations. The blue bytes in the forward chunk can be determined by the previously chosen blue bytes in the initial structure.

The white bytes in the forward chunk stands for the bytes whose value are affected by the red bytes from initial structure and can't be precomputed until the partial match is found. Similarly, in the backward chunk, red and white cells stand for the certain and uncertain bytes. The gray cells are constant bytes in the target value, the chaining value and the initial structure.

Since this is a second preimage attack, the second last chaining value and the last message block with proper padding are known. We choose random messages and get a random chaining value at the third last position. With this chaining value, apply MitM preimage attack of the compression function.

With chunk separation in Fig. 14, we have a MitM attack with $D_1 = 72, D_2 = 64, m = 64$ and $b = 512$. According to equation 4, the complexity can be computed as $2^{-1}2^{512}(2^{-72}+2^{-64}+2^{-64}) \approx 2^{448}$. Memory requirement is 2^{64}. Note that the complexity of computing the two chunks and checking the full match may be different. Here, we don't consider the difference between them and they are all regarded as the same cost of half compression function call. Methods for calculating freedom degrees of two chunks and the size of matching point are described in appendix A, which can also be found in [14].

C.3 First Preimage Attack on `Whirlpool`

This attack consists of three steps: First, find a preimage of the last block with proper padding. Second, construct an expandable message. At last, connect expandable message and the last block with MitM. The attack process is illustrated in Fig. 15.

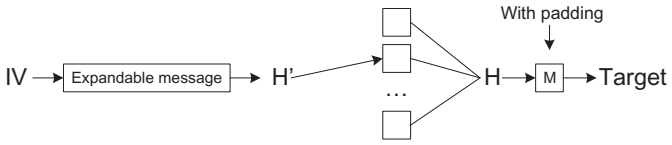

Fig. 15. Outline of the first preimage attack on 5-round `Whirlpool`

Dealing with Message Padding In order to apply the first preimage attack, the message padding must be dealt with properly. In our attack, the last message block consists of 255-bit message concatenated with one bit of "1" padding and 256-bit binary expression of the message length l. Since `Whirlpool` uses 512-bit message block, $l \equiv 255 \mod 512$. Then the last 9 bits of l are fixed to 011111111.

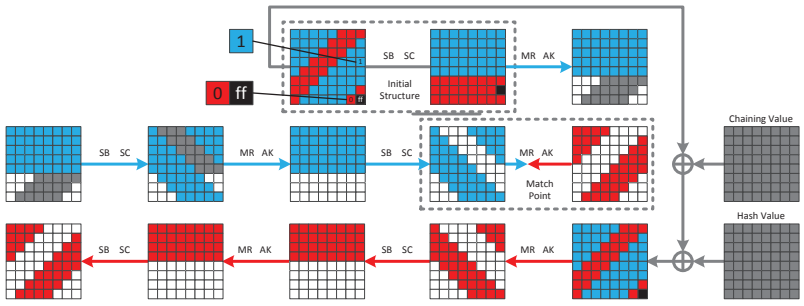

Fig. 16. Chunk separation for the last message block of `Whirlpool`

In Fig. 16, the initial structure is relocated at the beginning of the compression function for the convenience of the message padding, since in MP mode, the first state is the message block itself. Value of the black byte in the first state is fixed to $0xff$, because it is the last 8 bits of l. One red byte is marked with a "0", which means the last bit of it is fixed to zero due to the message length l. There is another blue byte marked with a "1", which comes from the "1-0" padding.

Parameters for this MitM attack are $D_1 = D_2 = 63, m = 64$ and $b = 512$. According to equation 4, the complexity is about 2^{449} computations and 2^{63} memory for the last block. When the attack on the last block is done, the remaining bits of the message length are fulfilled by expandable messages.

Expandable Messages Expandable messages [10] can be constructed using either Joux's multi-collision [9] or fix points of the compression function.

Expandable 2^k-collision can be constructed with $k \cdot 2^{n/2}$ computations and k memory. But its length can only be in the range of $[k, k + 2^k - 1]$ blocks. If the message length obtained from the last block is less than k (with a very small probability), we choose different random constants and repeat the attack.

Fix points of MP mode can be constructed by finding the zero preimages of the compression function in MMO mode, since

$$W_H(M) \oplus M \oplus H = H \Leftrightarrow W_H(M) \oplus M = 0.$$

This can be done using the same technique as in our 2nd-preimage attack, with complexity of $(2^{449}, 2^{64})$, which is an affordable cost for us. Note that for random H, the fix point exists with probability of $1 - e^{-1}$. If no fix point can be found for IV, we choose a random message block, compute the following chaining value and try to find fix point for this chaining value instead.

So, either way is fine to construct the expandable message here and has little influence on the overall complexity.

Turns Pseudo Preimage into Preimage After preparing preimage for the last message block $F(H, M) = T$ and the expandable message $IV \xrightarrow{M^*} H'$. Now we can connect them to form a first preimage.

Suppose it takes 2^c to find a pseudo preimage. A traditional MitM approach can convert preimage attack on the compression function into preimage attack on the hash function works like this. First, find and store 2^k pseudo preimages with 2^{k+c} computations and 2^k memory. Then choose 2^{n-k} random message , calculate from IV to find a chaining value appearing in one of the pseudo preimages. The complexity is $2^{n-k} + 2^{k+c}$. Take the optimal $k = \frac{n-c}{2}$, the minimum complexity is $2^{\frac{n+c}{2}+1}$. Using the pseudo preimage attack described in Sect. C.2, the preimage attack has a complexity of $(2^{481.5}, 2^{64})$.

D Improved MITM Attacks against AES Hashing Modes

It has been shown by Sasaki [14] that pseudo/second preimage of 7-round AES can be found in 2^{120} under hashing modes of Davies-Meyer (DM), Matyas-Meyer-Oseas (MMO), and Miyaguchi-Preneel (MP). In this section, we show that the pesudo-preimage can speed up with the help of the multi-target pseudo-preimage techniques proposed in [7], when the number of given targets is more than one. These improvements result in faster preimage and second preimages of 7-round AES hash modes. Details of the results are summarized in Table 5, comparing with those by Sasaki in [14].

Attack	Mode	Time	Memory	Message length	Reference
2nd Preimage	MMO, MP	2^{120}	2^8	-	[14]
	MMO, MP, DM	2^{128-k}	2^k	2^k blocks	[10]
	MMO, MP, DM	$2^{120-min(k,24)}$	$2^{8+min(k,24)}$	2^k blocks	this section
Preimage	DM	2^{125}	2^8	-	[14]
	DM	$2^{122.7}$	2^{16}	$> 2^8$ blocks	this section

Table 5. Results on (Second) Preimages of 7-round AES Hashing Modes.

We refer to Fig. 17 for the details of the attack. The states of AES are divided into two chunks, while the backward (red) chunk consists of states #8—#15 and forward (blue) chunk consists of states #20—#28 and #0—#7, the initial structure works for states #16—#19. These divisions are same as in [14]. However, when setting degree freedoms for both chunks, we find that we can have 2^{32} and 2^8 for the backward and forward directions, respectively. There is only one free byte in blue as in state #15 and the rest three bytes in the colunn are set to some constants. This byte is later propagated through the MixColumn into all four bytes of the first column in state #16. Similarly, we do not allow influence from red bytes in state #19 into the blue bytes in #20. Hence there is only (3-2)=1 free byte in each column of #19, which results in at most 2^{32} (4 bytes) for backward chunk. If one considers the situation that there are 2^k available targets T. Then the attack works in the follows.

1. Use $2^{8+min(k,24)}$ freedom degrees out of 2^{32} for the backward direction, and compute the values of the bytes in red and gray from state #15 back to #8, store them in a table.
2. For all 2^8 freedom degrees, compute the values of the bytes in blue and gray from state #15—#28, then for each of the 2^8 candidates, xor the $2^{min(k,24)}$ targets, so that $2^{8+min(k,24)}$ candidates will be available for state #0. Continue compute forward up to state #7.
3. Carry out the indirect partial matching between state #7 and #8.
4. Repeat until a full match is found.

It is easy to see that the overall complexity for this attack is $2^{120-min(k,24)}$, with memory requirement $2^{8+min(k,24)}$. While this can be directly applied when finding second preimages, we will use a tree-like construction as in [7] to find a first preimages for the AES hash in Davies-Meyer mode. Generally, given k targets with $1 < k < 2^{24}$, a pseudo preimage can be found in $2^{120}/k$. When finding the first preimage given one target T, one finds a pseudo preimage with chaining T_2 in 2^{120}, then finds the second pseudo preimage with the target set $\{T, T_2\}$ in time $2^{120}/2$, and so on. Hence finding Z pseudo preimages costs $\sum_{z=1}^{Z} 2^{120}/z \simeq 2^{120} \cdot \ln(Z)$. Finally, finding a message linking the IV to one of the targets costs $2^{128}/Z$. The overall time complexity is $2^{120} \cdot \ln(Z) + 2^{128}/Z$, which is $2^{122.7}$ when $Z = 2^8$ is chosen.

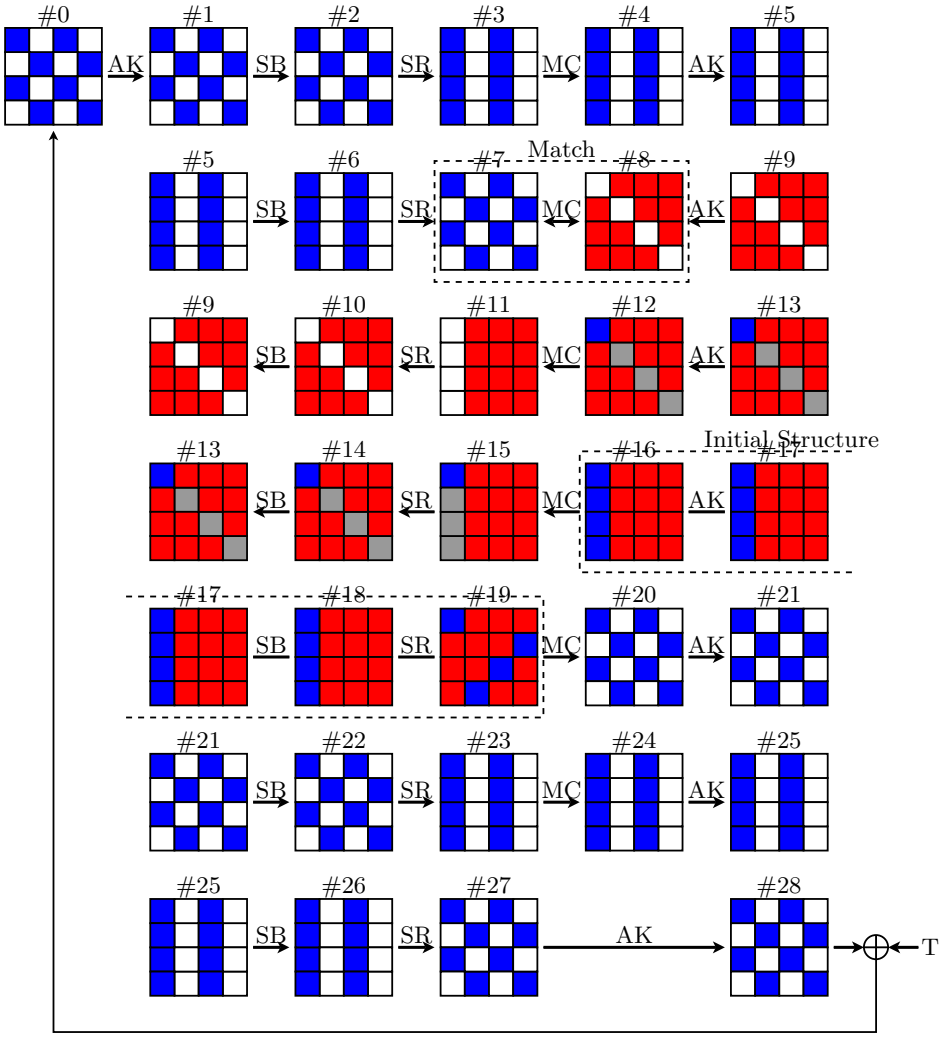

Fig. 17. MITM Attack against 7-Round `AES`