Improved (Pseudo) Preimage Attack and Second Preimage Attack on Round-Reduced Grøstl

Jian Zou, Wenling Wu, Shuang Wu, and Le Dong

Institute of Software
Chinese Academy of Sciences
Beijing 100190, China
{zoujian, wwl, wushuang, dongle}@is.iscas.ac.cn

Abstract. Grøstl is one of the five finalists in the third round of SHA-3 competition hosted by NIST. In this paper, we use many techniques to improve the pseudo preimage attack on Grøstl hash function, such as subspace preimage attack and guess-and-determine technique. We present improved pseudo preimage attacks on 5-round Grøstl-256 and 8-round Grøstl-512 respectively. The complexity of the above two attacks are $(2^{239.90}, 2^{240.40})$ (in time and memory) and $(2^{499.50}, 2^{499})$ respectively. Furthermore, we propose pseudo preimage attack and pseudo second preimage attack on 6-round Grøstl-256. The complexity of our 6-round pseudo preimage and second preimage attack is $(2^{253.26}, 2^{253.67})$ and $(2^{251.0}, 2^{252.0})$ respectively. As far as we know, these are the best known attacks on round-reduced Grøstl hash function.

Key words: Grøstl, meet-in-the-middle, guess-and-determine, initial structure, preimage attack

1 Introduction

Cryptographic hash functions are an important primitive in the modern cryptography; they are used in many applications, including MAC algorithms, digital signature schemes and so on. In general, hash function must satisfy three security requirements: preimage resistance, second preimage resistance and collision resistance. After the pioneering work of Wang, many new methods were proposed to attack the hash function in the last few years. There is a strong need for a secure and efficient hash function since the cryptanalysis of hash functions has been significantly improved. In 2008, NIST announced the SHA-3 competition[8] in order to select a secure hash function design.

Grøstl[4] is one of the five finalists in the third round of SHA-3 competition, which is proposed by Gauravaram *et al.* Grøstl is based on a wide-pipe compression function that is iterated in an MD-style manner. The compression function of Grøstl uses the SPN structure that follows the AES design strategy. Since the state is double-sized as the hash value, it's difficult to find an attack on Grøstl hash function. The Grøstl hash function has been modified in the third round. The old version is renamed to Grøstl-0 and the new one is called Grøstl-1.

In FSE 2008, Leurent[7] proposed the first preimage attack on the full MD4 hash function. From then on, many techniques are proposed to improve the preimage attacks. One of them is the meet-in-the-middle (MitM) preimage attack with the splice-and-cut technique. This method is first proposed by Aoki and Sasaki to attack MD4[2]. The MitM attack preimage attacks have been applied to many hash function such as HAVAL-3/4[10], MD4[7, 2], MD5[11], Tiger[5], RIPEMD[15], SHA-0/1[3], and SHA-2[1, 5]. In CRYPTO 2009, Aoki and Sasaki[3] combined the MitM attack with the guessing of carry bits technique to improve their preimage attack on SHA-0 and SHA-1. Then Sasaki[9] proposed the MitM preimage attack on AES hash mode for the first time in FSE 2011. In FSE 2012, Wu *et al.*[16] improved its complexity and proposed the first pseudo preimage attack on Grøstl.

Our contributions In [16], Wu *et al.* converted the preimage attack on Grøstl into a three-sum problem. By using the b-bit partial preimage attack, Wu *et al.* made their complexity lower than the birthday bound. In this paper, we find out it is not necessary to solve the b-bit partial preimage attack. We propose the subspace preimage attack to replace the b-bit partial preimage attack. We will introduce the subspace preimage attack in Section 3.

Using the subspace preimage attack, we propose an improved 5-round pseudo preimage attack on Grøstl-256. Then we combine the guess-and-determine technique with the MitM attack to improve the 8-round pseudo preimage attack of Grøstl-512. We also propose a pseudo preimage attack and a pseudo second preimage attack on 6-round Grøstl-256 hash function by using a complicated initial structure. Our cryptanalytic results of Grøstl are summarized in Table 1. We will explain these results in Section 4, 5 ,6 and 7.

Table 1. Comparison to previous works

Algorithm	Target	Attack Type	Rounds	Time	Memory	Source
Grøstl-256	Hash Function	Collision	3	2^{64}	-	[13]
	Compression Function	Semi-Free-Start Collision	6	2^{112}	2^{64}	[13]
	Permutation	Distinguisher	8	2^{48}	2^{8}	[12]
	Output Transformation	Preimage	5	2^{206}	2^{48}	[16]
	Output Transformation	Preimage	6	2^{251}	2^{251}	[6]
	Output Transformation	Preimage	6	2^{246}	2^{8}	Sect.6.1
	Hash Function	Pseudo Preimage	5	$2^{244.85}$	$2^{230.13}$	[16]
	Hash Function	Pseudo Preimage	5	$2^{239.9}$	$2^{240.4}$	Sect.4
	Hash Function	Pseudo Preimage	6	$2^{253.26}$	$2^{253.67}$	Sect.6
	Hash Function	Pseudo Second Preimage	6	2^{251}	2^{252}	Sect.7
Grøstl-512	Hash Function	Collision	3	2^{192}	-	[13]
	Compression Function	Semi-Free-Start Collision	6	2^{152}	2^{56}	[12]
	Output Transformation	Preimage	8	2^{495}	2^{16}	[16]
	Output Transformation	Preimage	8	2^{470}	2^{120}	Sect.5.1
	Hash Function	Pseudo Preimage	8	$2^{507.32}$	$2^{507.00}$	[16]
	Hash Function	Pseudo Preimage	8	$2^{499.5}$	2^{499}	Sect.5

Outline of the paper This paper is organized as follows. We describe Grøstl hash function in Section 2. In Section 3, we make a brief introduction to the previous work and our improvement work on Grøstl. Then we give our attacks on Grøstl in Section 4, 5, 6 and 7. Section 8 concludes the paper.

2 Specifications and Notations

2.1 Specification of Grøstl

Grøstl adopts a wipe-pipe design. The chaining value are 512-bit and 1024-bit for Grøstl-256 and Grøstl-512 respectively. Message length should be less than $2^{73} - 577$ bits. The padding rule is omitted here, since it's not important in our attack. Assume a message is padded and divided into message blocks $M_0, M_1, \ldots, M_{k-1}$. The hash value h is generated as follows:

$$\begin{cases} CV_0 \leftarrow IV \\ CV_{i+1} \leftarrow CF(CV_i, M_i) \text{ for } i = 0, 1, \ldots, k-1 \\ h = Trunc(P(CV_k) \oplus CV_k) \end{cases}$$

Here the IV is the initial value, and $CF(CV_i, M_i)$ is the compression function of Grøstl. $P(CV_k)$ is a permutation that is a component of the compression function and will be defined later.

The compression function uses two permutations $P(\cdot)$ and $Q(\cdot)$, and computes as follows(see also Fig.1):

$$CF(CV_i, M_i) = P(CV_i \oplus M_i) \oplus Q(M_i) \oplus CV_i.$$

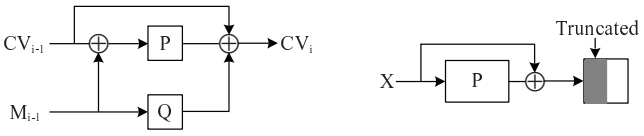

Fig. 1. Compression function and output transformation of Grøstl

0	8	16	24	32	40	48	56
1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63

Fig. 2. Byte positions for Grøstl-256

Here, byte positions in a state S for Grøstl-256 are denoted by integer numbers B, $B \in \{0, 1, 2, \ldots, 63\}$, as shown in Fig. 2. We often denote several bytes of state S by $S[a, b, \ldots]$, e.g. 8 bytes in the right most column are denoted by $S[56, 57, \ldots, 63]$. CV_i is the chaining value and M_i is a message block. $P(\cdot)$ and $Q(\cdot)$ are AES-like permutation with 8×8 and 8×16 sized state for Grøstl-256 and Grøstl-512 respectively. Grøstl-256 adopts 10-round $P(\cdot)$ and $Q(\cdot)$. Grøstl-512 uses 14-round $P(\cdot)$ and $Q(\cdot)$. The round function of the permutations consists of the following four operations:

- **AddConstant:** The AddConstant operation XORs a round-dependent constant to the state.
- **SubBytes:** The SubBytes transformation applies an S-box to each cell of the state.
- **ShiftRows:** The ShiftRows transformation cyclically rotates the cells of the i-th row leftwards by shift vector (define later).
- **MixColumns:** In the MixColumns operation, each column of the matrix is multiplied by an MDS matrix.

We use AC, SB, SR and MC to denote these transformations for short. The shift vectors used in $P(\cdot)$ and $Q(\cdot)$ are different. $P(\cdot)$ in Grøstl-256 uses (0,1,2,3,4,5,6,7) and $P(\cdot)$ in Grøstl-512 uses (0,1,2,3,4,5,6,11). $Q(\cdot)$ in Grøstl-256 uses (1,3,5,7,0,2,4,6) and $Q(\cdot)$ in Grøstl-512 uses (1,3,5,11,0,2,4,6). For a detailed explanation, we refer to the original paper[4].

As shown in the submission document of Grøstl hash function[4], if we make $CV_i^{'} = CV_i \oplus M_i$, then compression function can be rewritten as

$$CF(CV_i, M_i) = P(CV_i^{'}) \oplus CV_i^{'} \oplus Q(M_i) \oplus M_i.$$

This important property will be used in our attack on Grøstl.

2.2 Definition of Attacks

Suppose that the target function is H with a range of $\{0,1\}^n$. Several attacks are defined as follows.

Definition 1 (Preimage attack). *Given IV and t, find M such that $H(IV, M) = t$.*

Definition 2 (Pseudo preimage attack). *Given t, find (IV, M) such that $H(IV, M) = t$.*

Definition 3 (Partial preimage attack). *Given t_0, find M such that $H(M) = t_0||*$ (or $Trun_k(H(M)) = t_0$), where t_0 is a k-bit value and $Trun_k()$ is the truncation function.*

Definition 4 (Subspace preimage attack). *Given t, find M such that $L(H(M)) = t$, where $L()$ is a function with a range of $\{0,1\}^k$ and $k \leq n$.*

Definition 5 (Pseudo second preimage attack). *Given t, IV_1, M_1, find (IV_2, M_2) such that $H(IV_2, M_2) = H(IV_1, M_1) = t$.*

The partial preimage attack are used in Wu *et al.*[16] for the intermediate step of the pseudo preimage attack on the hash function. In this paper, our 5-round and 8-round pseudo preimage attacks for Grøstl hash function are based on subspace preimage attack. Specifically, by finding preimages in a linear subspace (with a linear function L), we can reduce the complexity of the previous attack. The details will be described in the following sections.

3 Previous Work and Our Improvement

3.1 Previous Work

In FSE 2012, Wu *et al.*[16] proposed the first pseudo preimage attack on Grøstl hash function. They idea can be summarized as below. Suppose the hash output is n-bit and the state size is 2n-bit. To find a pseudo preimage (CV, M) of Grøstl, it is desirable to invert the output transformation of Grøstl. Let $X = CF(CV, M)$, then X is the preimage of the output transformation. With $H^{'} = CV \oplus M$, we get

$$(P(H^{'}) \oplus H^{'}) \oplus (Q(M) \oplus M) \oplus X = 0.$$

Then the pseudo preimage attack turns into a three-sum problem.

Using four parameters x_1, x_2, x_3 and b, the attack process can be described as follow(see also in Fig.3):

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 the leftmost b bits of $P(H^{'}) \oplus H^{'}$ are zero. This step is considered to find partial zero preimages on $P(H^{'}) \oplus H^{'}$. Then store $P(H^{'}) \oplus H^{'}$ and $H^{'}$ in lookup table L_2.

3. Find 2^{x_2} random M with the correct padding and calculate $Q(M) \oplus M$. Check if there exists an entry in L_1 that matches the leftmost b bits of $Q(M) \oplus M$. Then $2^{x_1+x_2-b}$ partial matches $Q(M) \oplus M \oplus X$, whose leftmost b bits are all zero, is expected to remain.
4. For each of the $2^{x_1+x_2-b}$ $Q(M) \oplus M \oplus X$ remained in step 3, check whether there exists an entry in L_2 that matches the remaining $(2n-b)-bits$. Once a full match is found, a pseudo preimage of Grøstl hash function is found.

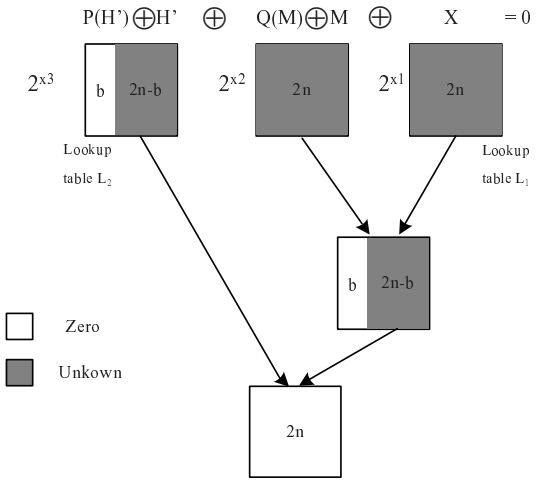

Fig. 3. Outline for the Wu *et al.* attack on Grøstl

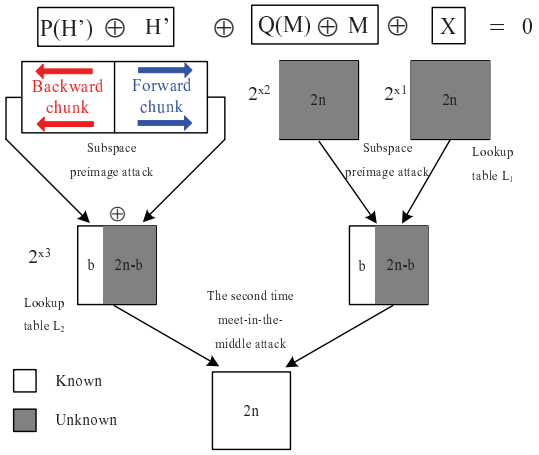

Fig. 4. Outline for our pseudo preimage attack on Grøstl

Suppose for Grøstl with $2n$-bit state, it needs $2^{C_1(2n,n)}$ computations to find a fixed position n-bit partial preimage of X and it needs $2^{C_2(2n,b)}$ computations to find a chosen position b-bit partial preimage of $P(X) \oplus X$. Then the complexity for each step can be calculated as follows:

1. In Step 1, building the lookup table L_1 takes $2^{x_1+C_1(2n,n)}$ computations and 2^{x_1} memory.
2. In Step 2, building the lookup table L_2 takes $2^{x_3+C_2(2n,b)}$ computations and 2^{x_3} memory.
3. In Step 3, It takes 2^{x_2} Q calls to calculate $Q(M) \oplus M$ in order to check the partial match in lookup table L_1. The complexity is equivalent to 2^{x_2-1} compression function calls.
4. In Step 4, checking the final match for $2^{x_1+x_2-b}$ candidates needs $2^{x_1+x_2-b}$ table look-ups, which can be equivalently regarded as $2^{x_1+x_2-b} \cdot C_{TL}$ compression function calls. C_{TL}[1] is the complexity of one table lookup, where unit one is one compression function call. C_{TL} is chosen as 1/640 and 1/2048 for 6-round Grøstl-256 and 8-round Grøstl-512 respectively.

And the overall complexity to compute the pseudo preimage of Grøstl is:

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

with memory requirement of $2^{x_1} + 2^{x_3}$. Note that the formula of $2^{x_1+x_2+x_3-2n} \geq 1$ should be satisfied in order to find one final match.

[1] The constant C_{TL} is the upper bound of the complexity that one table lookup takes. Consider the fact that 5-round Grøstl-256 software implementation composes of (8*8)*5*2=640 s-box lookups, and 8 round Grøstl-512 composes of (8*16)*8*2=2048 s-box lookups.

3.2 Outline for Our Pseudo Preimage Attack on the Grøstl Hash Function

We find out there is no need to find a b-bit chosen position partial preimage. Here we use the subspace preimage attack to improve their algorithm. We show an easy example of the subspace preimage attack in Fig.5. We should compute the 16 bits state value(red) in partial preimage attack, however we just need to find two state value(red) to satisfy a given 8-bit linear relationship(decided by the blue states) in the subspace preimage attack. Note that the subspace preimage attack is valid when the whole attack process can be divided into two phases MitM attack(see Fig.4). Then we will match the left relationships in the second time MitM attack.

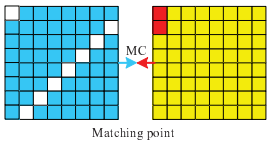

Fig. 5. an easy example of subspace preimage attack

In [16], the notation $2^{C_2(2n,b)}$ stands for the complexity to find a b-bit chosen position partial preimage of $P(H^{'}) \oplus H^{'}$. Here we adopt the same notation for the subspace preimage. Here, b denotes the bit size of the output of the linear function L. The linear subspace defined by the equation $L(x) = t_0$ has a dimension of $n - b$. Note that a b-bit partial preimage is a special kind of subspace preimage, where the linear function for the subspace is $L(x) = Trun_b(x)$.

As shown in Fig.4, the attack can be described as follows: we adopt the subspace preimage attack in the first stage of MitM attack. Then we match the left bytes in the second time MitM attack.

Using the same four parameters x_1, x_2, x_3 and b, our attack process can be described as follow:

1. Find 2^{x_1} preimages X of the output transformation and store them in a lookup table L_1.
2. Find 2^{x_3} subspace preimage of $P(H^{'}) \oplus H^{'}$. Then store $P(H^{'}) \oplus H^{'}$ and $H^{'}$ in a lookup table L_2.
3. Find 2^{x_2} random M with the correct padding and calculate $Q(M) \oplus M$. Check if there exists an entry in L_1 that $X \oplus Q(M) \oplus M$ is in the same subspace of $P(H^{'}) \oplus H^{'}$. Then $2^{x_1+x_2-b}$ partial matches $Q(M) \oplus M \oplus X$, are expected to remain.
4. For each of the $2^{x_1+x_2-b}$ $Q(M) \oplus M \oplus X$ remained in step 3, check whether there exists an entry in L_2 that matches the remaining $(2n - b)$ bits. Once a full match is found, a pseudo preimage of Grøstl hash function is found.

Then the overall complexity to compute the pseudo preimage of Grøstl is similar to Wu *et al.*[16] and can be written as:

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

with memory requirement of $2^{x_1} + 2^{x_3}$. Note that our subspace preimage attack just improve the $2^{C_2(2n,b)}$.

4 Improved Pseudo Preimage Attack on 5-round Grøstl-256

According to equation(1), the overall complexity to compute the pseudo preimage of Grøstl depends on these parameters: $x_1, x_2, x_3, C_1(2n, n)$ and $C_2(2n, b)$. If we can reduce the complexity of $2^{C_1(2n,n)}$ and $2^{C_2(2n,b)}$, the overall complexity can be reduced. As a result, we try to minimize the complexity of $2^{C_1(2n,n)}$ and $2^{C_2(2n,b)}$.

Since Wu *et al.*[16] have proved that their method to compute $2^{C_1(512,256)} = 2^{206}$ for 5-round Grøstl-256 is optimal, it's impossible to improve the $2^{C_1(512,256)}$. However we find that $2^{C_2(512,b)}$ can be reduced by the subspace preimage attack. As a result, we can improve the overall complexity of pseudo preimage attack on 5-round Grøstl-256.

4.1 Subspace Preimage Attack on $P(H') \oplus H'$ (Reduce the $2^{C_2(2n,b)}$)

We show the chunk separation for the subspace preimage attack in Fig.6.

Parameter for the subspace MitM attack As shown in Fig.6, we adopt the MitM attack here. There are five colors shown in the attack. The red/blue color means neutral message. They are independent from each other. The white color stands for the bytes whose values are affected by red bytes and blue bytes both, and we can't determine their values until a partial match is found. The gray bytes are constants that we can choose randomly in the initial structure. The yellow bytes are truncated bytes.

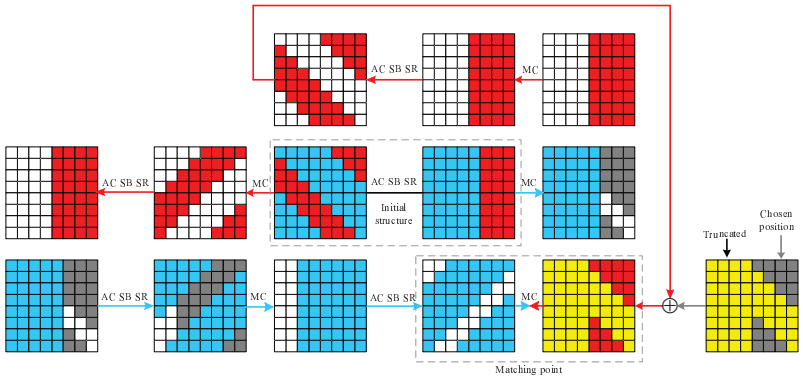

Fig. 6. Chunk separation of 5-round chosen position partial preimage attack on $P(H') \oplus H'$ for Grøstl-256

In order to compute the complexity for the MitM attack, some parameters for the attack should be presented: the matching point size m. Here we use D_r and D_b to denote the freedom degrees for the red and blue bytes respectively. Without loss of generality, we assume that $D_b \geq D_r$. We use d_r and d_b to indicate the actually used freedom degrees:

The MitM Attack Algorithm The MitM preimage attack can be described in the following steps:

1. Set random value to constants in the initial structure.
2. For all possible values 2^{d_r} of red bytes, compute backward from the initial structure and obtain the value at the matching point. Store the values in a lookup table L_r.
3. For all possible values 2^{d_b} of blue bytes, compute forward from the initial structure and obtain the result at the matching point. Check if there exists an entry in L_r that matches the result at the matching point.
4. If no full match has been found, repeat the above 3 steps.

In Fig.6, freedom degrees for the red and blue bytes are $D_r = 48$ bits and $D_b = 64$ bits respectively. Besides the matching point size $m_{max} = 64$ bits, since we can construct 64 bits linear relationship for the last four columns. If we just set $d_r = d_b = 48$ bits and $m_{best} = 48$ bits, we can find $2^{48} (= 2^{48+48}/2^{48})$

48-bit subspace preimages with the complexity of 2^{48}. It means that a 48-bit subspace preimage is found with the complexity of $2^1 (= 2^{48}/2^{48})$. Here the complexity is measured by compression function calls. It takes about half $P(\cdot)$ calls in the MitM attack, i.e. 1/4 compression function calls to evaluate the matching point for one direction. So we multiply 2^{-2} to the complexity: $2^{C_2(512,48)} = 2^{-2} \cdot 2^1 \approx 2^{-2}$. We will multiply the same coefficient in the following of this paper. Then our $2^{C_2(512,48)} \approx 2^{-2}$ is smaller than the $2^{C_2(512,31)} \approx 2^{14}$ in Wu *et al.* attack.

Minimizing the Overall Complexity With $2^{C_1(512,256)} = 2^{206}$ and $2^{C_2(512,48)} = 2^{-2}$, we can compute the best overall complexity of pseudo preimage attack for 5-round Grøstl-256. When $b = 48, x_1 = 32.4, x_2 = 239.4$ and $x_3 = 240.4$, we can improve the overall complexity to about $2^{239.9}$. Memory requirement is about $2^{240.4}$.

5 Improved Pseudo Preimage Attack on 8-round Grøstl-512

Here we talk about how to improve the $2^{C_1(1024,512)}$ and $2^{C_2(1024,b)}$ to reduce the overall complexity for 8-round pseudo preimage attack on Grøstl-512 hash function.

5.1 Preimage Attack on 8-round Grøstl-512 Output Transformation (Reduce the $2^{C_1(1024,512)}$)

Our 8-round preimage attack on Grøstl-512 output transformation is obtained by combining guess-and-determine technique with the MitM attack. The guess-and-determine technique is first used in the preimage attack on SHA-0/SHA-1 by Aoki and Sasaki[3]. By guessing some unknown bytes, all the possible values of the guessed bytes are used as extra freedom degrees in the MitM attack. As a result, we can obtain enough matching points. Note that, the guessed bytes are extra constraints after a partial match is found, and we should check the guessed value to produce a valid preimage. We will talk about more details about the guessing technique in the following attack algorithm. The chunk separation for the 8-round attack(combine guess and determine with the MitM) is shown in Fig.7. Since Grøstl-512 compression function has slow diffusion, we can construct a 2-round initial structure easily.

Parameters for the Guess-and-Determine and the MitM Attack As shown in Fig.7, we use the purple bytes as the guessed bytes and the other colors are the same meanings as described in the MitM attack. In order to evaluate the complexity for the attack, we should define these parameters: freedom degrees in red and blue bytes (D_r, D_b), the guessing red and blue bytes (D_{gr}, D_{gb}), the bits of the matching point m and the bits of the partial preimage size b. Here we also use (d_r, d_b) to denote the actually used freedom degrees. Note that this attack is different from the simple MitM attack so that (d_r, d_b) don't satisfy the equation (2).

The Attack Algorithm and Complexity The attack algorithm is similar to the MitM attack and can be described as follows:

1. Set the constants in the initial structure randomly.
2. For all possible values 2^{d_r} of the red bytes and $2^{D_{gr}}$ of the guessing red bytes, compute backward from the initial structure and obtain the value at the matching point. Store the values in a lookup table L.
3. For all possible values 2^{d_b} of the blue bytes and $2^{D_{gb}}$ of the guessing blue bytes, compute forward from the initial structure and obtain the value at the matching point. Check if there exists an entry in L that matches the result at the matching point. Expected number of the partial matches is $2^{d_r+d_b+D_{gr}+D_{gb}-m}$.

4. Once a partial match is found, compute and check if the guessed value is right. The probability of the validity is $2^{-D_{gr}-D_{gb}}$. There are $2^{d_r+d_b-m}$ valid partial matches left. Then we continue the computation and check the full match. The probability that a partial match is a full match is $2^{-(n-m)}$.
5. The success probability for the above steps is $2^{d_r+d_b+D_{gr}+D_{gb}-m} \cdot 2^{-(D_{gb}+D_{gr})} \cdot 2^{-(n-m)} = 2^{d_r+d_b-n}$. Then repeat the above steps for $2^{n-d_b-d_r}$ to find one full match.

The complexity for each step can be calculated as follows:

1. In Step 2, building the the lookup table L takes $2^{d_r+D_{gr}}$ computations and memory.
2. In Step 3, it takes $2^{d_b+D_{gb}}$ computations to find the partial matches. Expected number of the partial matches is $2^{d_b+D_{gb}+d_r+D_{gr}-m}$.
3. In Step 4, testing all the partial matches in step 3 needs $2^{d_b+D_{gb}+d_r+D_{gr}-m}$ computations. The probability of the validity is $2^{-D_{gb}-D_{gr}}$, and there are $2^{d_b+d_r-m}$ valid partial matches left.
4. In Step 5, repeat the above four step for $2^{n-d_b-d_r}$ times.

Then the complexity of the above attack algorithm is:

$$2^{n-d_b-d_r} \cdot (2^{d_r+D_{gr}} + 2^{d_b+D_{gb}} + 2^{d_b+D_{gb}+d_r+D_{gr}-m}) = 2^n \cdot (2^{D_{gb}-D_r} + 2^{D_{gr}-d_b} + 2^{D_{gb}+D_{gr}-m}). \quad (2)$$

As shown in Fig.7, the parameters for the attack on the output transformation of 8-round Grøstl-512 are as follow:

The parameters for the attack are as follows: $d_r = d_b = 80$ bits, $D_{gr} = D_{gb} = 40$ bits, $m = 144$ bits and $n = 512$ bits. According to equation(2), the complexity is $2^{C_1(1024,512)} = 2^{-2} \cdot 2^{512} \cdot (2^{40-80} + 2^{40-80} + 2^{40+40-144}) \approx 2^{470}$ compression function calls and $2^{40+80} = 2^{120}$ memory. Then our $2^{C_1(1024,256)} \approx 2^{470}$ is smaller than the $2^{C_1(1024,256)} \approx 2^{495}$ in Wu *et al.* attack. Note that the guess-and-determine technique is especially useful for the slow diffusion target.

5.2 Subspace Preimage Attack on $P(H') \oplus H'$ (Reduce the $2^{C_2(1024,b)}$)

We show chunk separation in Fig.8. Note that we don't adopt the guess-and-determine technique here, because we can't find a good guess-and-determine method to make the complexity better than the simple MitM attack. We find out the guess-and-determine method is not suit for the b-bit subspace preimage attack(when b is small).

The parameters for the MitM attack: the freedom degrees $D_r = 16$ bits, $D_b = 16$ bits. The size of matching point $b_{max} = 32$ bits. We set $d_r = d_b = b_{best} = 16$ bits, then we can find $2^{16} (= 2^{16+16}/2^{16})$ 16-bit subspace preimage with the complexity of 2^{16}. It means a 16-bit subspace preimage is found with the complexity of $2^1 (= 2^{16}/2^{16})$. Then the complexity is $2^{C_2(1024,16)} = 2^{-2} \cdot 2^1 \approx 2^{-2}$.

Minimizing the Overall Complexity According to equation (1) and $2^{C_1(1024,512)} = 2^{470}, 2^{C_2(1024,16)} = 2^{-2}$, we can rewrite the overall complexity as $2^{x_1+470} + 2^{x_3-2} + 2^{x_2-1} + 2^{x_1+x_2-16} \cdot C_{TL}$. When $b = 16$, $x_1 = 27, x_2 = 498$, and $x_3 = 499$, the overall complexity is the lowest:$\approx 2^{499.5}$. Memory requirement is 2^{499}.

Note that, we can use the memoryless MitM attack in our pseudo preimage attack, but the memoryless MitM attacks lead to higher complexity. Moreover, the memory require of our attack is mainly on the generalized birthday attack [14] step, which is much bigger than in the MitM preimage attacks. As a result, we don't use the memoryless MitM techniques.

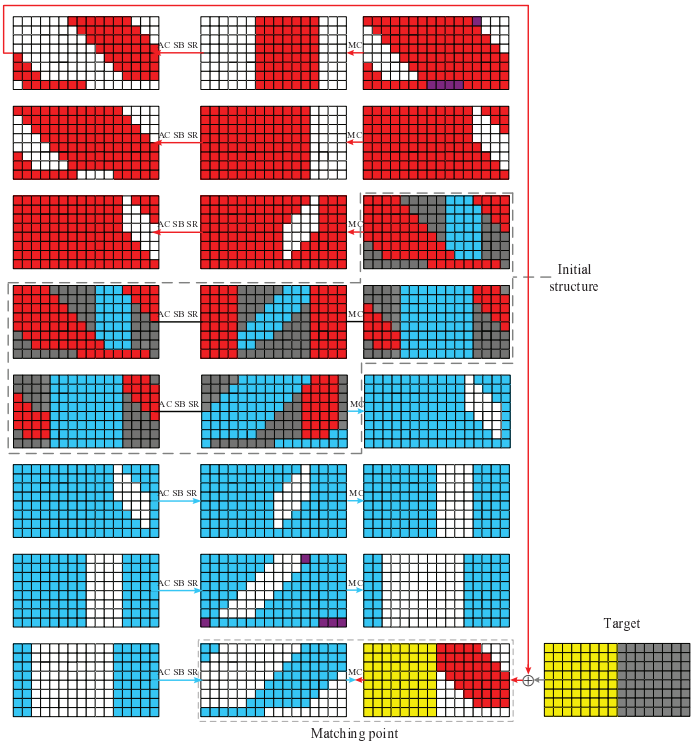

Fig. 7. Chunk separation of 8-round fixed position partial preimage attack on $P(X) \oplus X$

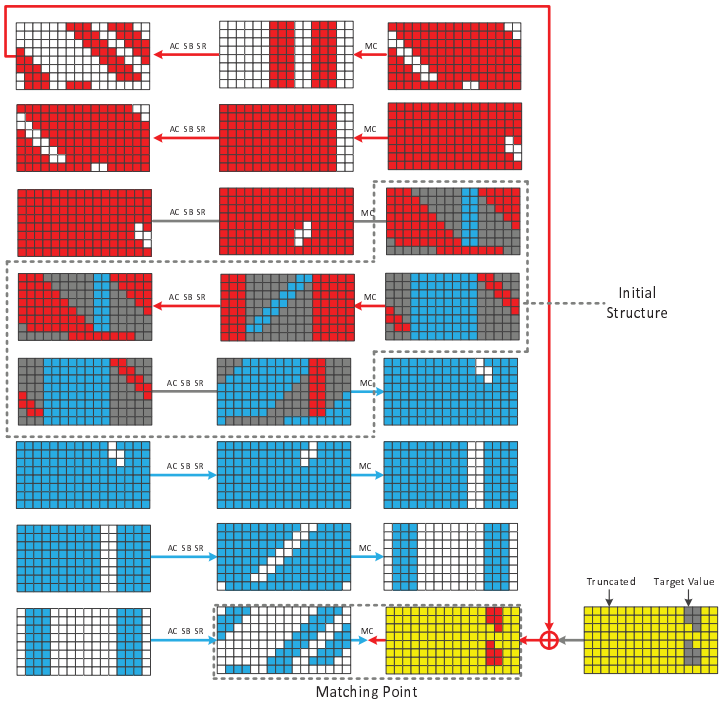

Fig. 8. Chunk separation of 8-round chosen position partial preimage attack on $P(H^{'}) \oplus H^{'}$

6 Pseudo Preimage Attack on 6-round Grøstl-256 Hash Function

Since two rounds of Grøtl-256 compression function achieves the full diffusion, the previous attack without any new techniques are at most five rounds of Grøtl-256. Khovratovich[6] has proposed a 6-round attack on Grøtl-256 output transformation function using biclique technique.

6.1 Preimage Attack on 6-round Grøstl-256 Output Transformation

Our 6-round preimage attack is obtained by combining the MitM attack with a complicated initial structure. The detail of the initial structure is shown as follows(also in Fig.9):

1. Randomly choose the constant values for the State #6$[1, 3, 5, 7, 8, 10, 12, 14, 17, 19, 21, 23, 24, 26, 28, 30]$ (gray).
2. Set the value of the State #4$[0, 5, 6, 7, 8, 9, 14, 15, 16, 17, 18, 23, 24, 25, 26, 27]$ (blue) so that the chosen 4 bytes at State #3$[7, 14, 21, 28]$ (red) can be achieved through the InverseMixColumns operation.
3. For each value of the State #4$[0, 5, 6, 7, 8, 9, 14, 15, 16, 17, 18, 23, 24, 25, 26, 27]$ (blue), we calculate the correspondent values of the State #5$[0, 1, 2, 3, 5, 6, 7, 15, 16, 17, 22, 23, 24, 29, 30, 31]$ (blue) through the SubBytes and ShiftRows operations.
4. With the known values of State #5 (blue) and State #6 (gray), we calculate the rest blue bytes of State #5 and State #6 through the MixColumns operation.
5. With the calculated blue bytes of State #5$[4, 5, 6, 7, 11, 12, 13, 14, 18, 19, 20, 21, 25, 26, 27, 28]$, we compute the values of the State #4$[33, 34, 35, 36, 42, 43, 44, 45, 51, 52, 53, 54, 60, 61, 62, 63]$ through the InverseShiftRows and InverseSubBytes operations. We check whether the values of the State #4$[33, 34, 35, 36, 42, 43, 44, 45, 51, 52, 53, 54, 60, 61, 62, 63]$ satisfy the liner relationship so that the chosen 4 bytes at #3 (#3$[35, 42, 49, 56]$) can be achieved through the InverseMixColumns operation.

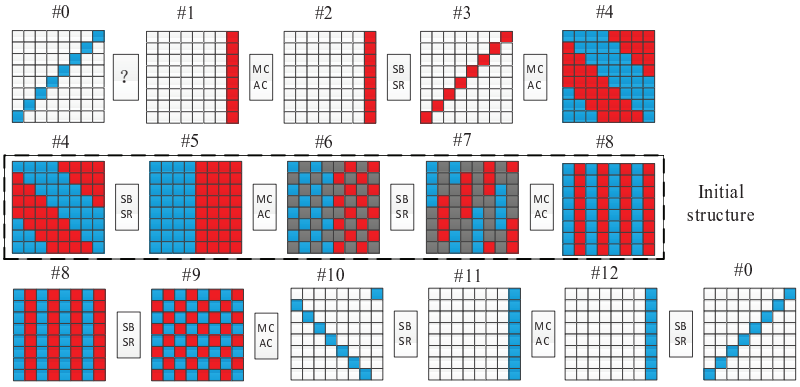

Fig. 9. Chunk separation of output transformation of Grøstl-256

Suppose for Grøstl-256 compression function, it needs 2^{C_r} computations to find a suitable initial structure for the forward direction(from #6 to #8) and it needs 2^{C_b} computations to find a suitable initial structure for the backward direction(from #6 to #4). Then the complexity to find a suitable initial structure can be calculated as follows:

$$2^{n-d_b-d_r} \cdot (2^{d_r+C_r} + 2^{d_b+C_b} + 2^{d_b+d_r-m}) = 2^n \cdot (2^{C_b-d_r} + 2^{C_r-d_b} + 2^{-m}). \quad (3)$$

Note that initial structure is from State #4 to State #8, the above steps just show the process from State #6 to State #4 (backward direction). The process from State #6 to State #8 (forward direction) is similar to the process from State #6 to State #4, and we just omit the details. In step 5, State #4[33, 34, 35, 36, 42, 43, 44, 45, 51, 52, 53, 54, 60, 61, 62, 63] should satisfy a 8 bit liner relationship in each column, so we should repeat the above 5 steps 2^{32} times to get a initial structure that meet our requirements.

Compared to guess-and-determine MitM attack, our new attack has two advantages:

1. We don't require the matching points $m > D_b + D_r$. It only requires $d_r > C_b$ and $d_b > C_r$.
2. There is no guessed bytes, we don't have to check if the guessed values is right. So this attack is suitable for the partial preimage attack.

The parameters for the attack are as follows: $C_r = C_b = 32$ bits, $d_r = d_b = 40$ bits, $m = 8$ bits and $n = 256$ bits. According to equation (3), the complexity is $2^{C_1(512,256)} = 2^{-2} \cdot 2^{256} \cdot (2^{32-40} + 2^{32-40} + 2^{-8}) \approx 2^{246}$ compression function calls. Note that our (2^{246},2^8)(time,memory) is smaller than (2^{251},2^{251}) in Khovratovich attack[6].

6.2 Preimage Attack on $P(H') \oplus H'$

Since the guess-and-determine technique is not suitable for the partial preimage attack, Our new 6-round preimage attack is suitable to achieve a better $2^{C_2(2n,b)}$.

As shown in Fig.9, the parameters for our attack: the complexity to construct the complicated initial structure $C_r = C_b = 32$ bits. The size of matching point $m = 8$ bits, We set $d_r = d_b = 40$ bits, then we can find $2^8 (= 2^{40+40-32-32}/2^8)$ 8-bit subspace preimage with the complexity of 2^8. It means a 8-bit subspace preimage is found with the complexity of $2^1 (= 2^8/2^8)$. Then the complexity is $2^{C_2(512,8)} = 2^{-2} \cdot 2^1 \approx 2^{-2}$.

Minimizing the Overall Complexity According to equation (1) and $2^{C_1(512,256)} = 2^{246}, 2^{C_2(512,8)} = 2^{-2}$, we can rewrite the overall complexity as $2^{x_1+246} + 2^{x_3-2} + 2^{x_2-1} + 2^{x_1+x_2-8} \cdot C_{TL}$. When $b = 8$, $x_1 = 5.67, x_2 = 252.67$, and $x_3 = 253.67$, the overall complexity is the lowest:$\approx 2^{253.26}$. Memory requirement is $2^{253.67}$.

7 Pseudo Second Preimage Attack on 6-round Grøstl-256 Hash Function

Suppose the hash size is n-bit and the state size is 2n-bit. We know

$$Grøstl(CV_0, M_0, M_1, \ldots, M_{k-1}, M_k) = h,$$

and we want to find another $(CV_0^{'}, M_0^{'}, M_1^{'} \ldots, M_{k-1}^{'}, M_k)$ such that

$$Grøstl(CV_0^{'}, M_0^{'}, M_1^{'} \ldots, M_{k-1}^{'}, M_k) = h.$$

Let $CV_k = CF(CV_{k-1}, M_{k-1})$, then CV_k is the input to the last block of compression function. If we want to find the pseudo second preimage of Grøstl, then CV_k and M_k is known to us. Note that M_k obeys the right padding rule. If we can use a technique to find $(CV_{k-1}^{'}, M_{k-1}^{'})$ that satisfies $P(CV_{k-1}^{'} \oplus M_{k-1}^{'}) \oplus Q(M_{k-1}^{'}) \oplus CV_{k-1}^{'} = CV_k$, then we can also use the technique to find $(CV_{k-2}^{'}, M_{k-2}^{'}) \ldots (CV_0^{'}, M_0^{'})$ such that $Grøstl(CV_0^{'}, M_0^{'}, M_1^{'} \ldots, M_{k-1}^{'}, M_k) = h$. We can find a pseudo second preimage of Grøstl, after we repeated the technique k times. In the following, we will talk about how to find $(CV_{k-1}^{'}, M_{k-1}^{'})$ to satisfy the formula.

With $H^{'} = CV_{k-1}^{'} \oplus M_{k-1}^{'}$, we have $(P(H^{'}) \oplus H^{'}) \oplus (Q(M_{k-1}^{'}) \oplus M_{k-1}^{'}) = CV_k$.

As shown in Fig.10, the attack turns into a two phases MitM attack. With parameters y_1, y_2, y_3 and b, the attack process can be described as follows:

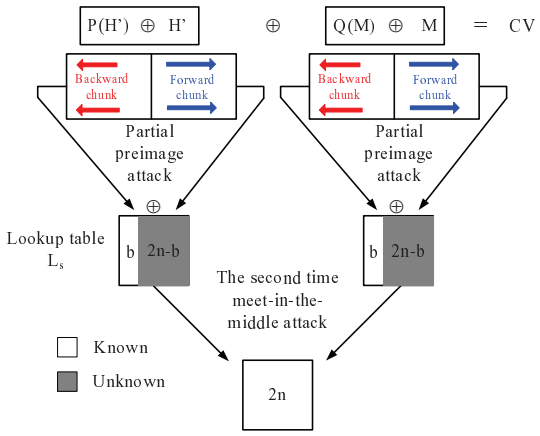

Fig. 10. Outline for our pseudo second preimage attack on Grøstl

1. Find 2^{y_1} $b-bit$ partial preimages of $P(H^{'}) \oplus H^{'}$. Here we need to store the 2^{y_1} $P(H^{'}) \oplus H^{'}$ and $H^{'}$ in the lookup table L_s.
2. Find 2^{y_2} the same $b-bit$ partial preimage of $Q(M^{'}_{k-1}) \oplus M^{'}_{k-1}$ as $P(H^{'}) \oplus H^{'}$ in L_s. Check if the remaining $2n-b$ bits are the same to CV_k.

Once a full match is found, we find a $(CV^{'}_{k-1}, M^{'}_{k-1})$ such that $P(CV^{'}_{k-1} \oplus M^{'}_{k-1}) \oplus Q(M^{'}_{k-1}) \oplus CV^{'}_{k-1} = CV_k$. Then we can repeat the above attack algorithm k times to find a pseudo second preimage of Grøstl.

Suppose that for Grøstl with 2n-bit state, it needs $2^{C_3(2n,b)}$ and $2^{C_4(2n,b)}$ computations to find a $bbit$ partial preimage of $P(H^{'}) \oplus H^{'}$ and $Q(M^{'}_{k-1}) \oplus M^{'}_{k-1}$ respectively. Then we can calculate the complexity for each step of the attack algorithm:

1. In Step 1, it takes $2^{y_1+C_3(2n,b)}$ computations and 2^{y_1} memory to build the lookup table L_s.
2. In Step 2, it takes $C_4(2n,b)$ computations to calculate the same $b-bit$ partial preimage of $Q(M^{'}_{k-1}) \oplus M^{'}_{k-1}$ as $P(H^{'}) \oplus H^{'}$. Calculating 2^{y_2} $b-bit$ partial preimage of $Q(M^{'}_{k-1}) \oplus M^{'}_{k-1}$ and checking the remaining $2n-b$ partial match in table L_s.

So the overall complexity is: $2^{y_1+C_3(2n,b)} + 2^{y_2+C_4(2n,b)}$, with memory requirement of 2^{y_1}. Note that we need $2^{y_1+y_2-2n} \geq 1$ in order to find one full match.

7.1 Subspace Preimage Attack On $P(H^{'}) \oplus H^{'}$ and $Q(M^{'}_{k-1}) \oplus M^{'}_{k-1}$ of Grøstl-256

The chunk separation for $P(H^{'}) \oplus H^{'}$ and $Q(M^{'}_{k-1}) \oplus M^{'}_{k-1}$ are shown in Fig.9 and Fig.11 respectively. As shown in Fig.9, the freedom degrees for the partial preimage attack on $P(H^{'}) \oplus H^{'}$ are $D_r = 128$ bits, $D_b = 128$ bits. The size of the matching point $b = 8$ bits and $n = 256$ bits. The complexity to construct the complicated initial structure $C_r = C_b = 32$ bits. We set $d_r = d_b = b = 40$ bits, then we can find $2^8 (= 2^{40+40-32-32}/2^8)$ 8 bit partial preimage with a complexity of 2^8. The average complexity to compute a 8 bit partial preimage is $2^1 (= 2^8/2^8)$. So we can calculate the complexity of a 8 bit partial preimage attack on $P(H^{'}) \oplus H^{'}$: $2^{C_3(512,8)} = 2^{-2} \cdot 2^1 \approx 2^{-2}$. This means we just need 2^{-2} computations to get a 8 bit linear relationship subspace preimage on $P(H^{'}) \oplus H^{'}$. As shown in Fig.11, it also takes 2^{-2} computations to get the same 8-bit partial preimage on $Q(M^{'}_{k-1}) \oplus M^{'}_{k-1}$ as $P(H^{'}) \oplus H^{'}$. Then we just need to choose $y_1 = y_2 = (512-8)/2 = 252$, the complexity to calculate the pair $(CV^{'}_{k-1}, M^{'}_{k-1})$ is $2^{252-2} + 2^{252-2} \approx 2^{251}$ and the memory is 2^{252}. We just need to repeat the attack k times to get the

pseudo second preimage of Grøstl-256. Since the guess-and-determine method is not suit for the b-bit subspace preimage attack(when b is small), we can't find a pseudo second preimage attack on 8-round Grøstl-512 that its complexity is better than the pseudo preimage attack on 8-round Grøstl-512.

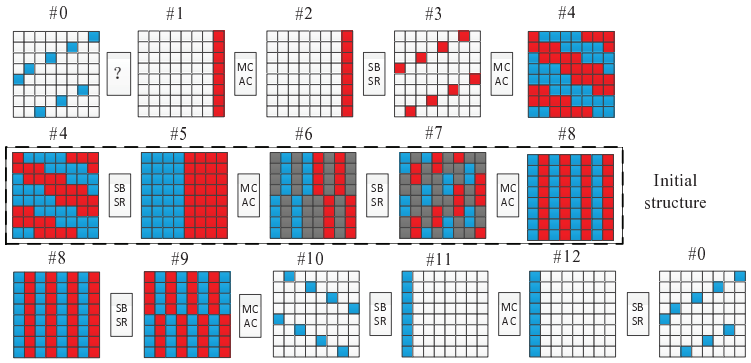

Fig. 11. Chunk separation of $Q(M'_{k-1}) \oplus M'_{k-1}$ of Grøstl-256

8 Conclusion

In this paper, first we improve the pseudo preimage attack on 5-round Grøstl-256. Besides we improve the 8-round Grøstl-512 by using the guess-and-determine technique during the MitM attack. At last we construct 6-round pseudo preimage attack and 6-round pseudo second preimage attack on Grøstl-256 by using a complicated initial structure.

We found out the guess-and-determine is useful to solve the full preimage attack instead of the partial preimage attack. It takes some complexity to check if the guessing bits is correct, which is not suitable to solve the partial preimage problem. Since the choices of number and position for the guessing bytes are too many, it seems impossible to do an exhaustive search. So it remains to be an open problem to find the optimal guess-and-determine strategy for Grøstl.

Besides the subspace preimage attack is just to replace the partial preimage attack. If we can divide the attack process into several times MitM attack, we can use the subspace technique to improve the complexity.

In our attack , we have difficulty in attacking more rounds of Grøtl because the difference ShiftRow operations between $P(\cdot)$ and $Q(\cdot)$. These result show it's good to adopt difference ShiftRow operations between $P(\cdot)$ and $Q(\cdot)$. Our attacks do not threat any security claims of Grøtl.

Acknowledgement The authors would like to thank Lei Wang and Jian Guo for their inspiring suggestions.

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. 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 (Round 3), 2011.
5. 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.
6. Dmitry Khovratovich. Bicliques for permutations: Collision and preimage attacks in stronger settings. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 544–561. Springer, 2012.
7. Gaëtan Leurent. MD4 is Not One-Way. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *LNCS*, pages 412–428. Springer, 2008.
8. 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).
9. 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.
10. 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.
11. 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.
12. 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.
13. Martin Schläffer. Updated Differential Analysis of Grøstl. Grøstl website, January 2011.
14. David Wagner. A Generalized Birthday Problem. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.
15. 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 Aggelos Kiayias, editor, *CT-RSA*, volume 6558 of *LNCS*, pages 197–212. Springer, 2011.
16. Shuang Wu, Dengguo Feng, Wenling Wu, Jian Guo, Le Dong, and Jian Zou. (pseudo) preimage attack on round-reduced grøstl hash function and others. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 127–145. Springer, 2012.