(Chosen-multi-target) preimage attacks on reduced Grøstl-0

Sareh Emami, Praveen Gauravaram**, Josef Pieprzyk and Ron Steinfeld

Macquaire University, Australia and CR RAO AIMSCS India*

Abstract The cryptographic hash function Grøstl is a finalist in the NIST's SHA-3 hash function competition and it is a tweaked variant of its predecessor called Grøstl-0, a second round SHA-3 candidate. In this article, we consider 256-bit Grøstl-0 and its 512-bit compression function. We show that internal differential trails built between the two almost similar looking permutations of the compression function can be coverted to chosen-multi-target-preimage attacks, a variant of multi-target preimage attacks. Consequently, we show chosen-multi-target-preimage attacks for up to 9 out of 10 rounds of the compression function and up to 7 rounds of the hash function. Finally, we use these attacks as a tool to find preimages and pseudo preimages for 6 rounds of the 256-bit Grøstl-0 hash function.

Keywords: Hash function, Compression function, Grøstl-0, Grøstl, (Chosen-multi-target) preimage attack, Super-Sbox attack.

1 Introduction

The cryptographic hash function Grøstl [3] is a finalist in the ongoing NIST's SHA-3 competition [14,13,15,8]. Grøstl [3] is an iterated hash function which operates in the wide-pipe hash function mode with an internal state of size at least twice of the hash value size. Its compression function uses two large distinct (fixed-key) permutations that have a byte-oriented substitution-permutation structure similar to AES [1]. Soon after its selection for the final round of the SHA-3 competition, a tweak was proposed for Grøstl and since then the earlier version has been referred to as Grøstl-0 [3] and SHA-3 finalist as Grøstl [4]. Extensive analysis of Grøstl-0 and its building blocks [10,9,11,6,16,5,18] has motivated the designers to tweak it by making its permutations to look more distinct for an enhanced security against attacks that apply to Grøstl-0 due to the usage of almost similar permutations. So far, these attacks on Grøstl-0 have employed the rebound attack [10] and its extensions such as Super-Sbox [7,5,11] and non-full-active Super-Sbox [18] attacks to find collisions for the reduced round compression function [10,9,5,18] or hash function [11,6] and to find distinguishers for the compression function [5,16] and permutations [9,5,16]. It is interesting to see whether the similar looking permutations in Grøstl-0 can be further exploited to mount preimage attacks or their variants on the compression function or hash function. We address this problem by analyzing 256-bit Grøstl-0 compression function and hash function against (chosen-multi-target) preimage attacks. When the l-bit permutations of Grøstl-0 are assumed ideal, the compression function has a $2^{l/2}$ security against preimage attacks [2,3]. Grøstl-0 with n-bit hash value has a claimed complexity of 2^n against preimage attacks [3]. These claims also apply to Grøstl.

Chosen-(multi-target preimage attacks). In a preimage attack on an n-bit hash function H, for a given y, an adversary finds a message M such that $H(M) = y$ in less than 2^n evaluations of H. Similarly, in a preimage attack on l-bit compression function cf, for a given y, an adversary finds an input chaining value and message block pair (h, m) such that $cf(h, m) = y$. The idea of multi-target (Aka one-of-many) preimage attacks on hash functions was first proposed by Merkle [12]. In this attack on an n-bit hash function H, an attacker aims to find a preimage for one of the given K

* A portion of this was done when the author was visiting Indian Statistical Institute, Kolkata, India.

Table 1: Summary of results on 256-bit Grøstl-0 where each attack requires 2^{64} memory.

Attack		Rounds	Targets	Time	Generic	Section
Comp. Function	Preimage	6	-	2^{128}	2^{256}	6
	Chosen-multitarget preimage	6	2^{64}	2^{64}	2^{224}	4.1
		6	2^{8}	2^{120}	2^{252}	4.1
		7	2^{80}	2^{64}	2^{216}	4.2
		7	2^{24}	2^{120}	2^{244}	4.2
		8	2^{192}	2^{64}	2^{160}	4.2
		8	2^{136}	2^{120}	2^{188}	4.2
		9	2^{192}	2^{120}	2^{160}	4.2
Hash Function	Preimage	5	-	2^{144}	2^{256}	6
		6	-	2^{144}	2^{256}	6
	Pseudo preimage	6	-	2^{128}	2^{256}	6
	Chosen-multi-target preimage	5	2^{64}	2^{80}	2^{192}	5.1
		6	2^{16}	2^{136}	2^{240}	5.1
	Chosen-multi-target pseudo preimage	6	2^{64}	2^{64}	2^{192}	5
		6	2^{8}	2^{120}	2^{248}	5
		7	2^{80}	2^{64}	2^{176}	5
		7	2^{24}	2^{120}	2^{232}	5

hash values in less than $2^n/K$ evaluations of H [12,17]. On a compression function cf, the task is to find a pair (h, m) which hits one of K outputs of cf in less than $2^n/K$ evaluations of cf. We call these attacks K-target preimage attacks. In a slightly different setting, an attacker may show the existence of a set of outputs for cf or H wherein a preimage can be found for one of the outputs in the set. We call attacks in this setting chosen-multi-target preimage attacks or chosen-K-target preimage attacks for an output set of size K.

Our analysis and results. We consider 256-bit Grøstl-0 which has a 10-round 512-bit compression function based on two almost similar 512-bit permutations and an output transformation based on 10 rounds of one of these permutations. We analyze reduced round variants of the compression function and hash function against (chosen-K-target) preimage attacks. To summarize:

1. In the ideal permutation model, we prove that finding a preimage for one among K-target compression function outputs has a complexity of about $2^{l/2}/\sqrt{K}$ queries to the permutations. While doing so, we also correct the previous security proof [2] which finds a security bound for the compression function against preimage attacks and we provide an accurate proof with almost similar security bound as [2]. These security bounds are also applicable to Grøstl compression function. We then present chosen-K-target preimage attacks for up to 9 rounds of the compression function for a complexity less than the generic attack complexity.
2. We convert some of the reduced-round chosen-K-target preimage attacks on the Grøstl-0 compression function into the corresponding chosen-K^*-target (pseudo) preimage attacks on the

hash function where $K^* \leq K$. As the K^* hash value targets of these attacks are determined by the K targets of the compression function attacks, we derive a generic security bound resembling a similar scenario in the ideal permutation model and compare our results with this bound. In addition, in some instances, we find reduced-round (pseudo) preimage attacks on Grøstl-0 where an attacker can find a preimage for any hash value challenged to the attacker from a set of hash values that are determined from the chosen targets of compression function attacks. As a consequence, we show 5 and 6-round (pseudo) preimage attacks on Grøstl-0 hash function.

In all our attacks we apply the idea of internal differences between similar looking permutations introduced for Grøstl-0 by Peyrin [16] to build new internal differential trails to find chosen-K-target preimages for the compression function. These trails are built by using an improved variant of the rebound attack called Super-Sbox attack which was used to find collisions for the reduced round Grøstl-0 compression function and hash function [11,16,5]. Table 1 summarizes our results.
Impact of our attacks. Our attacks highlight an interesting method of finding preimages for a specific set of hash values for reduced round Grøstl-0 by extending chosen-multi-target preimages built for its compression function. Although this property also holds for Grøstl due to its similar structure to Grøstl-0, more distinction between the permutations in Grøstl completely prevent the application of internal differentials even for very few rounds and therefore, any possibilities of building chosen-multi-target preimages for the compression function using internal differentials are ruled out. Therefore, our results strengthen the rationale for proposing the tweak for Grøstl-0 for the final round of the SHA-3 competition. In addition, our reduced-round preimage attacks on Grøstl-0 cannot be converted to finding second preimages as in our attacks a preimage can be found for any hash value from only a set of hash values.
Guide to the paper. In Section 2 we discuss 256-bit Grøstl-0 hash function; internal differentials for its compression function; the idea of rebound attack and its Super-Sbox extension. In Section 3 together with Appendices A and B, we derive generic security bounds for the Grøstl-0 compression function against (K-target) preimage attacks and for Grøstl-0 hash function against K^*-target preimage attacks respectively. Section 4 describes chosen-multi-target preimage attacks on several reduced round versions of the compression function. Chosen-multi-target (pseudo) preimage attacks and (pseudo) preimage attacks on reduced Grøstl-0 are shown in Sections 5 and 6 respectively. Finally, we conclude the paper in Section 7.

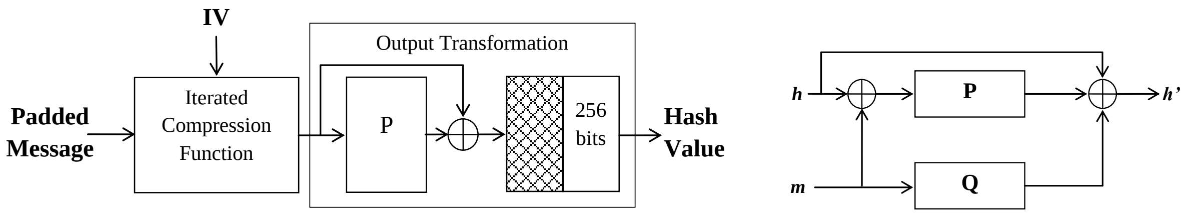
2 Background

2.1 Brief description of 256-bit Grøstl-0 hash function

The 256-bit Grøstl-0 hash function works as follows [3]: It takes an input message M and pads it securely and splits the padded message into equal length 512-bit blocks $m_1 \ldots m_t$. The initial value IV is defined as the 512-bit representation of the size of the hash value (i.e 256 bits) which in hexadecimal is 00 00 ... 01 00. Note that IV has only one non-zero byte. Each block is processed by iterating a compression function cf. At any iteration i, the compression function is defined by

$$cf(H_{i-1}, m_i) = P(H_{i-1} \oplus m_i) \oplus Q(m_i) \oplus H_{i-1} = H_i \qquad (1)$$

where H_{i-1} and H_i are the respective 512-bit input and output chaining values of cf, P and Q are 512-bit 10-round permutations each and $H_0 = IV$. The 512-bit output value H_t of the



(a) Abstract view of the hash function (b) Grøstl-0 compression function

Figure 1: Illustration of 256-bit Grøstl-0 hash function.

final compression function is processed by using an output transformation described by $OT(H_t) = \mathrm{trunc}_{256}(P(H_t) \oplus H_t)$ where the operation trunc_{256} discards all except the last 256 bits of $P(H_t) \oplus H_t$ to produce a hash value of 256 bits. An abstract view of 256-bit Grøstl-0 is illustrated in Figure 1. Often, in our analysis, we denote the pair (H_{i-1}, m_i) with (h, m) and H_i with h'.

The permutations P and Q are designed following the wide trail design strategy and in every round r_i for $i = 0, \ldots, 9$, they update an 8×8 state of 64 bytes. These transformations are similar to that of AES and are described as follows:

- AddRoundConstant (AC): Exclusive-or operation of distinct one byte constants to the 8×8 internal states of P and Q. For P, this constant is a round number i which is XORed to the first byte of the internal state. For Q, this constant is $i \oplus \mathrm{ff}$ XORed to the eighth byte of the internal state. The permutations P and Q differ in only this step.
- SubBytes (SB): Independent substitution of each byte in P and Q states by using AES S-box [1].
- ShiftBytes (SH): Cyclic left rotation of the j^{th} row state bytes by j positions where $j = 0, \ldots, 7$.
- MixBytes (MB): Linear diffusion layer in which each column of the state is multiplied with a constant 8×8 circulant MDS matrix.

The state S in a round r_i of the compression function is updated by a round transformation defined by $MC_i \circ SH_i \circ SB_i \circ AC_i(S)$ where $i \in \{0, \ldots, 9\}$. For more details of the design we refer to [3].

2.2 Rebound and Super-Sbox Attacks.

The rebound attack was proposed by Mendel *et al.* [10] to analyze block cipher and permutation based hash functions, in particular designs that are of AES-style or AES-based. On Grøstl-0, this attack decomposes the two permutations into three sub-permutations and employs identical truncated differentials between their round transformations in an *inside-out* approach via the *inbound* and *outbound* phases. The *inbound* phase starts with an 8-byte truncated difference between the MB layers of two consecutive rounds, say r_2 and r_3, and is propagated both forward and backward of the S-box layer of r_3. This phase is solved by an efficient meet-in-the-middle technique aided by the degrees of freedom called match-in-the-middle approach. The solutions of the *inbound* phase are pairs of values conforming the *inbound* phase differential trail which connects the input and output of S-boxes in r_3. Finally, these pairs are used as the starting points for the *outbound* phase which is the probabilistic part of the attack and extends the *inbound* phase differential trail in both the forward and backward directions.

The Super-Sbox attack [7,5] is an improved variant of the rebound attack to solve the *inbound* phase of two consecutive rounds covering two S-box layers and this method was applied to Grøstl-0 in [11]. When we apply this method to Grøstl-0, instead of checking each S-box for a valid differential as in the basic rebound attack, eight parallel 64-bit S-boxes called Super-Sboxes are checked. Figure 2 illustrates the *inbound* phase for a sample differential trail of the permutation P or Q of 256-bit Grøstl-0 between the state before the application of MB in round 1 and completion of round 3 (i.e between states S_1^{SH} and S_3^{MB}) and the first Super-Sbox is highlighted. Two adjacent SH and SB transformations in the first round of the *inbound* phase can be commuted without changing the final result. In Figure 2, every column in S_2^{SH} is the input to one 64-bit Super-Sbox and overall there are 8 independent Super-Sboxes. Although the differential distribution table (DDT) for the Super-Sbox includes 2^{128} entries; by fixing the input and output differences of the Super-Sbox, we can check all 2^{64} input values to see if the input difference to a Super-Sbox maps to the output difference.

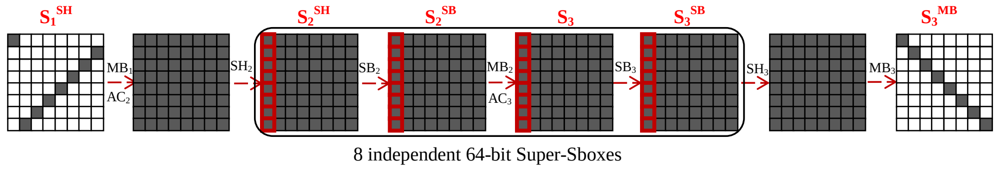


Figure 2: Application of Super-Sbox attack on Grøstl-0 compression function.

The inbound phase illustrated in Figure 2 is analyzed as follows:

1. For all 2^{64} differences at the state S_3^{MB}, compute backward through MB and SH to the state S_3^{SB}, then store the resulting 2^{64} differences for each column in list L_1.
2. Choose a random difference at S_1^{SH} and compute the difference forward to S_2^{SH}. Each column at state S_2^{SH} is the input to one Super-Sbox. For each Super-Sbox at state S_2^{SH}, connect the input and output differences as follows:

   (a) For the selected Super-Sbox (column) at state S_2^{SH}, try all 2^{64} possible pairs of values to calculate the output value of SB_2. Therefore, overall we can compute 2^{64} possible differences and corresponding pairs of values for the column at state S_2^{SB}.

   (b) Compute forward to S_3^{SB} through MB and SB, and find 2^{64} differences and corresponding pairs of values as the output of the Super-Sbox, then store them in list L_2.

   (c) Find a match between differences of list L_1 and corresponding differences and pairs of values in list L_2 through SB_3. Since each list has 2^{64} entries and we must match 64 bits (with the probability 2^{-64}) and find $2^{64} \times 2^{64} \times 2^{-64} = 2^{64}$ solutions for each Super-Sbox.

3. Since all Super-Sboxes are independent, find 2^{64} solutions for each Super-Sbox at S_3^{SB} and subsequently for the whole state with a complexity of 2^{64} time and memory. Hence, on average, the time complexity for each solution is one.

We can also choose 2^{64} differences for S_1^{SH} and repeat *inbound* phase with each of these differences to find up to 2^{128} solutions (pairs of values and differences) that satisfy the *inbound* phase trail. These pairs of values are the starting points for the probabilistic *outbound* phase of the attack. Memory required for any number of applications of the *inbound* phase is still 2^{64}.

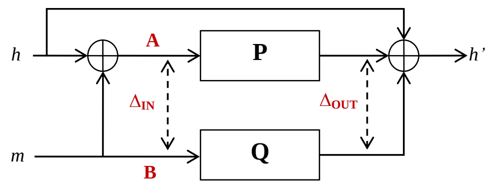


Figure 3: Illustration of internal differentials for Grøstl-0 compression function.

2.3 Internal differentials for 256-bit Grøstl-0

Peyrin [16] observed that when a cryptographic function is built upon similar parallel components then it may be possible to construct a differential trail which spans between these components and such a trail is called internal differential trail. Since the parallel permutations P and Q in 256-bit Grøstl-0 differ in only one byte of AC layer, internal differentials can be constructed for the compression function as illustrated in Figure 3. The differences between P and Q of Grøstl-0 compression function cf can be traced as follows: Let A and B be the input states for P and Q respectively. The input difference of the internal differential trail is given by $\triangle_{IN} = A \oplus B$ and its output difference by $\triangle_{OUT} = P(A) \oplus Q(B)$. Therefore, at any iteration i of cf, we can note that $h = A \oplus B$, $m = B$ and $h' = P(A) \oplus Q(B) \oplus A \oplus B = \triangle_{OUT} \oplus \triangle_{IN}$. The pair $(A, B) = (h \oplus m, m)$ is the valid pair confirming to the internal differential trail. Peyrin combined internal differentials with Super-Sbox attack to distinguish Grøstl-0 compression function from the one based on ideal permutations and to mount a collision attack on the 5 round compression function. This approach was extended by Ideguchi *et al.* [6] to find collisions for the 5 and 6 rounds of Grøstl-0 hash function. For details of the distinguisher and collision attacks we refer to [16,6].

3 The generic security of Grøstl-0 against (K-target) preimage attacks

For the compression function of Grøstl-0 with an l-bit output there is a security proof [2] which shows that when the permutations P and Q are assumed ideal, an adversary who makes at most q queries to these permutations and their inverses has an advantage of at most $q^2/2^l$ to find a preimage. That is, the complexity of a preimage attack on the compression function is about $2^{l/2}$ which is also the claimed security bound [3]. We remark that this security proof [2] is not completely accurate as it assumes that for an input query made to a permutation, the output is chosen uniformly at random from $\{0,1\}^l$ due to the random choice of the permutation. It seems that this proof does not take into consideration that P and Q are permutations. For example, after P has been queried i times on inputs $\alpha_1, \ldots, \alpha_i$, returning values $\beta_1 = P(\alpha_1), \ldots, \beta_i = P(\alpha_i)$, the response for the next query α depends on the previous responses $\beta_1, \ldots, \beta_i$ and hence, the returned value $\beta = P(\alpha)$ should be uniformly random over $2^l - i$ values (i.e all values except $\beta_1, \ldots, \beta_i$).

Therefore, in Appendix A we improve the security proof in [2] for Grøstl-0 compression function against preimage attacks and our security bound is almost the same as the one in [2]. We also extend our analysis to the security of the construction against K-target preimage attacks. To find a preimage for one of the K targets for the l-bit Grøstl-0 compression function requires about $2^{l/2}/\sqrt{K}$ queries. Since this analysis does not depend on the details of P and Q, it is reasonable to use its result for the chosen-K-target scenario as well. In Appendix B, we assume ideality of P and

Q and prove that K^*-target preimage attack on n-bit Grøstl-0 hash function is lower bounded by approximately $2^l/K$ queries, assuming that the target set Y is defined as $Y = OT(x) : x \in X$, for some fixed set X of compression function outputs (independent of P,Q) of size K. Note that this is the way Y is defined in our attacks in Section 5. Our results in Appendices A and B also apply to Grøstl compression function and hash function respectively as permutations are assumed ideal.

4 Chosen-K-target preimage attacks on 512-bit Grøstl-0 compression function

We first provide the main idea used in our chosen-K-target preimage attacks on the Grøstl-0 compression function followed by the details of the attacks in the subsequent sections.

In our attacks, we build internal differential trails between the permutations P and Q by using Super-Sbox attack so that the complexity of a chosen-K-target preimage attack is less than $2^{l/2}/\sqrt{K}$. An internal difference is the difference between P and Q. The size of K in our attacks is influenced by the factors such as the number of attackable rounds, the number of solutions obtained from the *inbound* phase of the Super-Sbox attack and the amount of control we would like to exert at the input state of the compression function. Recall from Section 2.3 that if A and B are inputs to the permutations P and Q, the compression function following the internal differential trail can be described as $cf(h, m) = h' = \triangle_{IN} \oplus \triangle_{OUT}$ where $\triangle_{IN} = A \oplus B$ and $\triangle_{OUT} = P(A) \oplus P(B)$.

Once the internal differential trail is built for a possible number of rounds, we take all possible outputs of this trail as the K-chosen-target set. Then we can find a preimage (h, m) for one of the K targets as follows: Since $A \oplus B = \triangle_{IN} = h$, when we extend the difference between the pair of values that satisfy the *inbound* trail and *outbound* trails of the Super-Sbox attack in the backward trail of the *outbound* phase, we would eventually end up with h and a pair of values (A, B) with the difference h. Note that the message block $m = B$ (i.e. the input value to permutation Q). By computing the forward trail of the *outbound* phase with the solution which satisfies both *inbound* and *outbound* phases, we end up hitting at least one of the K target hash values. A matched hash value is defined by $h' = \triangle_{IN} \oplus \triangle_{OUT}$. Thus, we have found the pair $(h, m) = (\triangle_{IN}, B)$ which hits one of the chosen-K-target hash values. If more than one solution satisfies the internal differential trail then we can find preimages for more than one K target outputs.

4.1 Attacks on 6-round compression function

Figure 4 shows the truncated internal differential trail for 6-round compression function where A and B are the inputs to the permutations P and Q and their difference $A \oplus B$ is the input chaining value h for the compression function. For this trail, we use the labels *controlled* and *uncontrolled* values to refer to the actual values of the states. *Controlled* values are known to the adversary from the beginning of the attack, for example in the state h' there are 56 *controlled* bytes and their values are fixed to zero. On the other hand, values of *uncontrolled* bytes are not known until the completion of the attack. Set Y is the target set which has 2^{64} 512-bit values. Every element of Y consists of only 8 *uncontrolled* bytes in the same positions as shown in the state h' in Figure 4. The application of Super-Sbox attack on these 6 rounds finds the preimage (h, m) of the compression function for a $h' \in Y$. The attack works as follows:

- *Inbound* phase: This phase starts at the state before MB_1 and ends before SB_4 as illustrated by the dashed lines in Figure 4. This phase is solved by the application of the Super-Sbox attack described in Section 2.2 which finds 2^{64} pairs which are also the starting points for the *outbound* phase. The complexity of this part is 2^{64} time and memory.

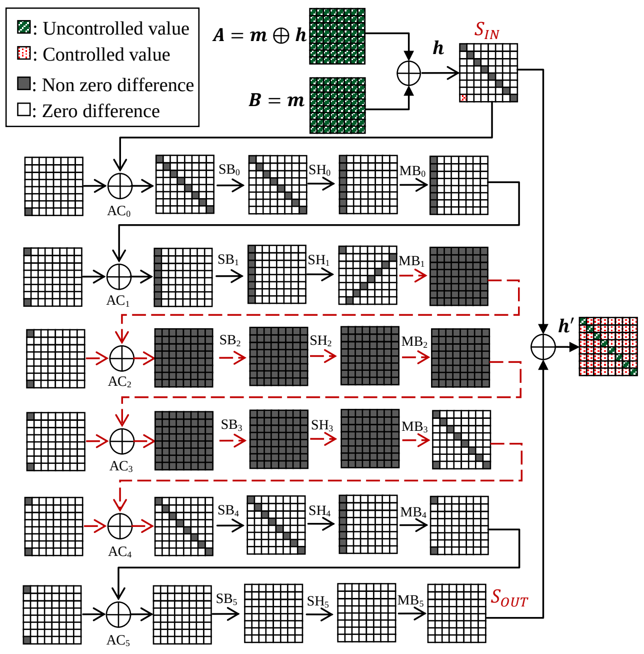


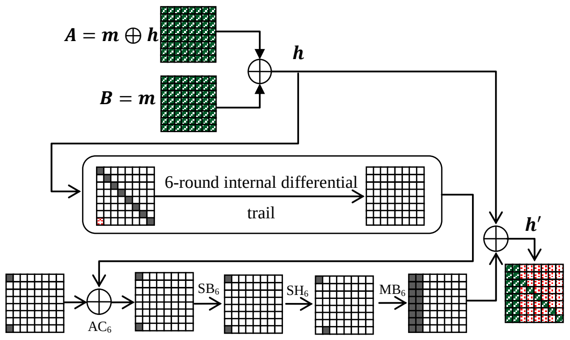
Figure 4: Internal differential trail for 2^{64}-target preimage attack on 6-round compression function.

- *Outbound* phase

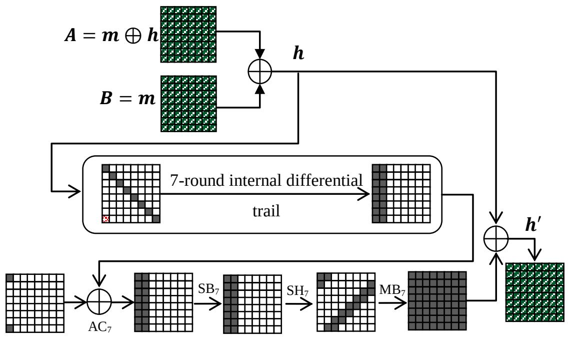
1. Use the starting points of the *inbound* phase to compute forward from the state before SB_4 to state S_{OUT}. This part is satisfied with 2^{-64} probability. Since, the $8 \mapsto 2$ bytes difference propagation through MB_4 is successful with 2^{-48} probability and two bytes differences are erased by AC_5 with 2^{-16} probability.
2. Use the corresponding starting point of the pair satisfying step 1 and work backward from the state before MB_1 to the input state S_{IN}. This step has a success probability of one.
3. The XOR addition of the input difference h at the state S_{IN} and output difference at S_{OUT} is h' which is one of the 2^{64} target output values of the compression function.
4. The pair of values that gives the difference at S_{IN} are the inputs of P and Q, their difference is the input chaining value h and the input value of Q is the message block m. Thus, we found the pair (h, m) for an output chaining value h' which is one of the target outputs.

The time complexity of the attack is 2^{64} computations due to the *outbound* phase and requires 2^{64} memory due to the *inbound* phase. This is much less than the generic complexity of 2^{224} to find a preimage for one among 2^{64} target compression function outputs.

Figure 7 in Appendix C illustrates another 6-round attack for 2^8 targets for a complexity of 2^{120} time (from *outbound* phase due to the $8 \mapsto 2$ difference propagation through MB_4, two bytes difference cancellation by AC_5 and $8 \mapsto 1$ difference propagation through MB_0) and 2^{64} memory (required for the *inbound* phase). This is much less than the generic attack complexity of 2^{252} for a 2^8-target preimage attack.



(a) 7-round internal differential trail for 2^{80} targets



(b) 8-round internal differential trail for 2^{192} targets

Figure 5: Extended internal differential trails of Figure 4 for chosen-K-target pseudo preimage attacks

4.2 Chosen-K-target preimage attacks on more rounds of the compression function

The internal differential trail of Figure 4 can be extended to 7 and 8-round trails as illustrated in Figure 5. As Figure 5a illustrates, by adding one round to the 6-round trail of Figure 4, we can generate the trail for 2^{80}-target preimage attack on 7-round compression function for no extra cost as the internal differentials of 7th round hold with a probability of one. MB_6 is a linear transformation which maps 2-byte differences to 16-byte differences. Therefore, for all 2^{16} possible differences in the state before MB_6, 2^{16} differences are generated for the state after MB_6. Since for every value among 2^{64} possibilities in h, there are 2^{16} values in the final state, there are up to 2^{80} possibilities for the state h'. This attack finds the chosen-multi-target preimage in 2^{64} time and memory, while the generic attack needs 2^{216} computations.

The 7-round trail of Figure 5a is extended by one more round to obtain an 8-round trail as illustrated in Figure 5b. Here MB_7 propagates the state of 16 non-zero differences to the full active state. However, due to the linearity of MB_7, there are 2^{128} possible values for the full active state. Since, there are 2^{64} possibilities for h, there are totally 2^{192} possible values for h' and hence 2^{192} targets for the compression function. The complexity of this attack is 2^{64} time and memory computations which is much less than 2^{160} generic attack complexity.

The 6-round internal differential trail of Figure 7 in Appendix C is also extended to 7 (Figure 8a) and 8-round (Figure 8b) internal differential trails to apply respectively 2^{24} and 2^{136}-target preimage attacks on the 256-bit Grøstl-0 compression function. Note that both extensions in the compression function rounds follow the outbound phase of 6-round attack explained in Section 4.1 with probability one. Therefore, the complexity of the 2^{24}-target preimage attack on 7-round compression function as well as the 2^{136}-target preimage attack on 8 rounds are 2^{120} time and 2^{64} memory.

2^{192}-target preimage attack on 9 rounds. The best possible internal differential trail in terms of number of rounds is a 9-round 2^{192}-target preimage attack illustrated in Figure 10 in Appendix C. The backward trail of the *outbound* phase is followed with 2^{-56} probability whereas the forward trail holds with 2^{-64} probability. Overall, the complexity of this attack is 2^{120} time and 2^{64} memory which is less than the generic complexity of 2^{160}.

5 Chosen-K^*-target (pseudo) preimage attacks on the reduced Grøstl-0

Sometimes, we can convert a chosen-K-target preimage attack on an r-round 256-bit Grøstl-0 compression function to the corresponding r-round chosen-K-target pseudo preimage attack on the hash function for a negligible additional complexity. The general description of this attack is outlined below:

1. Let X be the set of K target outputs of the r-round compression function where for one of which a preimage can be found. Choose a message block m' such that it satisfies the padding rule of Grøstl-0 and process it with the chaining values in the set X under the compression function to generate K outputs.
2. Process the above K outputs under the output transformation $OT(.)$ of Grøstl-0 to generate a set $Y = \{OT(cf(m', x)) : x \in X\}$ of K^* hash values.
3. Find the preimage (h, m) for a target in the set X. This implies that we have also found a pseudo preimage $(h, m\|m')$ for r-round Grøstl-0 for one of the hash values in the set Y. Note that the first compression function has r rounds whereas the second one may have all rounds.

We can use the above method to convert 6-round 2^{64} and 2^{8}-target preimage attacks on the compression function shown in Figures 4 and 7 to the corresponding 2-block chosen-multi-target pseudo preimage attacks on Grøstl-0 hash function respectively in 2^{64} and 2^{120} time complexity and 2^{64} memory in each of these attacks. In comparison, the complexity of generic attacks to find 2^{64} and 2^{8} target pseudo preimages is 2^{192} and 2^{248} respectively. Similarly, 7-round attacks illustrated in Figures 5a and 8a can be converted to the corresponding pseudo preimage attacks on the hash function for less than generic attack complexity. For beyond 7 rounds of 256-bit Grøstl-0, generic attacks are the best to find chosen-multi-target pseudo preimages.

5.1 Chosen-K^*-target preimage attacks on the reduced hash function

In the above attacks, if the input chaining value h of the first compression function is equal to IV then we obtain chosen-K^*-target preimage attacks on the hash function. Therefore, we present only the internal differential trails of chosen-K-target preimage attacks on the reduced compression function where $h = IV$ and then it is straight forward to convert them to the corresponding 2-block chosen-K^*-target preimage attacks for Grøstl-0.

Figure 6 illustrates internal differential trail to mount a 2^{64}-target preimage attack for the compression function with $h = IV$. The *inbound* phase of the attack starts at the 10-byte active state before MB_1 and ends at the 8-byte active state before SB_4. For a difference chosen at the state before MB_1, the *inbound* phase results in 2^{64} pairs of values in 2^{64} time and memory. Therefore, the *inbound* phase can produce up to $2^{64+80} = 2^{144}$ pairs of values as the starting points for the *outbound* phase in 2^{64} memory. The forward trail of the *outbound* phase holds with probability one. In the backward trail of the *outbound* phase, at first AC_1 cancels 2 active bytes in the first column of the state before SB_1 with 2^{-16} probability. The $8 \mapsto 2$ active bytes propagation through MB_0 has a success probability of 2^{-48}. Finally, with 2^{-8} probability, AC_0 cancels the one byte difference in the first column before SB_0 and with a probability of 2^{-8} the active byte of the input state is forced to be the non-zero byte in the IV of 256-bit Grøstl-0. Overall, the success probability of the *outbound* phase is 2^{-80}. Therefore, the total complexity of the attack is 2^{80} time and 2^{64} memory. Note that the complexity of the generic attack to find a preimage for one of 2^{64} targets is 2^{192}.

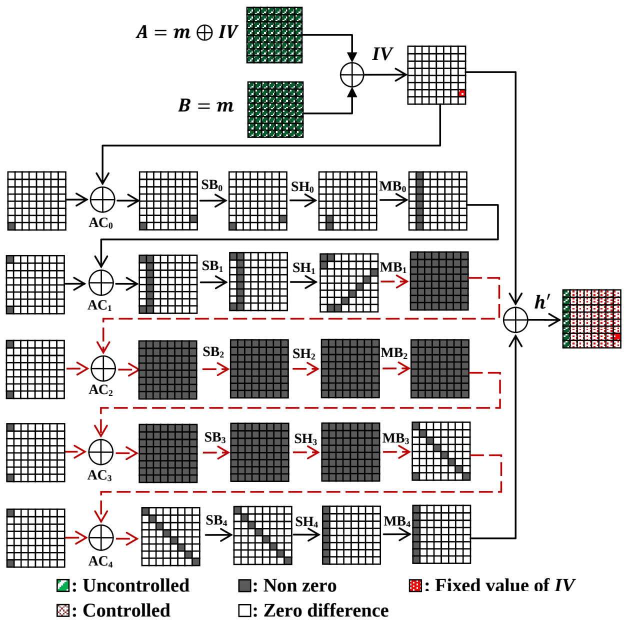


Figure 6: Internal differential trail for 2^{64}-target preimage attack on 5-round 256-bit Grøstl-0

This trail can be extended to 6 rounds as shown in Figure 9 in Appendix C to obtain a 2^{16}-target preimage attack. The extended round adds a factor of 2^{56} to the time complexity of the attack and makes it 2^{136} computations (the success probability of the forward outbound trail is 2^{-56} regarding $8 \mapsto 2$ propagation through MB_4 and one byte difference cancellation by AC_5). Also MB_5 is a linear function which transforms 2^{16} input possible differences to 2^{16} possible outputs, therefore there are 2^{16} values for h' and the hash targets.

Some remarks.

- Although truncation in $OT(.)$ may produce less than K target hash values of size K^* in the set Y, it does not decrease the importance of our attacks on the hash function because when the number of targets are reduced, the complexity of the generic attack increases. That is, $K^* \leq K$. Anyhow, our experiments show that the compression function and output transformation when evaluated with all possible 2^8 and 2^{16} chaining values as illustrated in the output state of Figures 7 and 9 respectively would also produce sets (Y) of hash values of the same size after the truncation step.
- In our chosen-K-target preimage attacks on the reduced round variants of the compression function from Section 4, the set of allowed outputs of size K that were committed to finding at the beginning of the attack is independent of P and Q since the values of the bytes in the non-zero positions of the output state in some attacks (e.g. Figure 4 and 6) and in the state before the final MB transform in other attacks (e.g. Figure 5) are allowed to be arbitrary. In the attacks on the reduced round variants of the hash function, the set of target hash values are determined by the target outputs of the first compression function which are computed independent of P and Q. However, the target hash value set is partially dependent on P and Q due to the computation of the second compression function and P in the output transformation.

Note that these attacks are distinct from the trivial scenarios where one can compute a set of hash values for some arbitrary messages and later show a preimage for one of the hash values. In Appendix B we show that a chosen-K-target preimage attack on Grøstl-0 compression function extends to a K^*-target preimage attack on hash function in at least $2^n/K$ complexity.

6 Preimage attacks on the reduced hash function and compression function

We can convert a chosen-K-target preimage attack on Grøstl-0 to a preimage attack if there is sufficient degrees of freedom available from the *inbound* phase of the attack. It is achieved by using the freedom to repeat the *inbound* phase and therefore the Chosen-K-target attack itself until we hit the challenged hash value. Hence, the 2^{64}-target preimage attack on 5-round hash function described in Section 5.1 can be converted to a preimage attack. Since there are 2^{64} targets according to the differential trails illustrated in Figure 6, we have to repeat the attack 2^{64} times to hit the desired hash value. The success probability of the attack depends on the *outbound* phase which is equal to 2^{-80}. Consequently, the overall complexity of the attack on the 5-round hash function is 2^{144} time and 2^{64} memory. Fortunately the *inbound* phase provides enough freedom for solving the *outbound* phase and repeating the attack to find the preimages. One may argue that we in fact do not find 2^{144} solutions for the inbound phase, because we use differentials to solve the *inbound* phase and it counts each pair of solution twice (e.g. there might be two pairs of (A, B) and (B, A) in the solutions). However, we do not consider this argument against our Super-Sbox attacks. Since we use internal differential trails between different permutations P and Q, two pairs of values in different orders result in different outputs after the *outbound* phase. Therefore, the *inbound* phase in Figure 6 generates 2^{144} starting points for the other parts of the attack. Note that by applying this attack we can find preimages of all the 2^{64} possible hash values. Similarly, we also extend the 2^{16}-target preimage attack on the 6-round hash function, illustrated in Figure 9 in Appendix C, to a preimage attack on 6 rounds. Since the complexity of the 2^{16}-target preimage attack is 2^{136} and we have to repeat the attack 2^{16} times to hit the desired hash value, the overall complexity of the preimage attack would be 2^{144} time and 2^{64} memory.

Likewise, the 2^{64}- and 2^8-target pseudo preimage attacks on the 6-round hash function, illustrated respectively in Figures 4 and 7, are extended to pseudo preimage attack on 6 rounds. In both attacks the overall complexity is 2^{128} time and 2^{64} memory. While in the former the adversary generates preimages for all 2^{64} target hash values.

In a preimage attack on a compression function cf, for a specific output h' we find a pair (h, m) such that $cf(h, m) = h'$. Obviously the chosen-multi-target preimage attacks on the 6-round compression function in Section 4.1, provide preimage attacks on the compression function. By repeating the 2^8-target preimage attack of Figure 7 or the 2^{64}-target preimage attack of Figure 4, we can obtain preimage attacks on the 6-round compression function in 2^{128} time and 2^{64} memory.

7 Conclusion

We used internal differential trails in the compression function of Grøstl-0 to present chosen-multi-target preimage attacks for up to 9 rounds of the compression function and extend some of these attacks to find chosen-multi-target (pseudo) preimages and preimages for the hash function. The best preimage attack was presented on 6 rounds of 256-bit Grøstl-0. The more distinct permutations used for Grøstl thwart internal differential attacks even for fewer rounds and hence our attacks do not apply to Grøstl. We also applied our analytical techniques on the 512-bit version of Grøstl-0 and we present these results in the full version of this paper. In future, we consider improving the complexities of our attacks by considering the improvements to the rebound attacks.

Acknowledgments: We are thankful to Florian Mendel and Christian Rechberger for discussions and comments on this paper.

References

1. Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer, 2002. 1, 2.1
2. Pierre-Alain Fouque, Jacques Stern, and Sébastien Zimmer. Cryptanalysis of Tweaked Versions of SMASH and Reparation. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 136–150. Springer, 2008. 1, 1, 3, A, A
3. Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl – A SHA-3 Candidate. Second Round of NIST's SHA-3 Competition, 2009. Available at `http://www.groestl.info/` (Accessed on 19/11/2010). 1, 2.1, 2.1, 3
4. Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl – A SHA-3 Candidate. A Finalist of NIST's SHA-3 Cryptographic Hash Function Competition, 2011. Available at `http://www.groestl.info/` (Accessed on 16/08/2011). 1
5. Henri Gilbert and Thomas Peyrin. Super-sbox cryptanalysis: improved attacks for AES-like permutations. In *Proceedings of the 17th international conference on Fast software encryption (FSE'10)*, Lecture Notes in Computer Science, pages 365–383. Springer, 2010. 1, 1, 2.2
6. Kota Ideguchi, Elmar Tischhauser, and Bart Preneel. Improved Collision Attacks on the Reduced-Round Grøstl Hash Function. In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, editors, *ISC*, volume 6531 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2010. 1, 2.3
7. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound Distinguishers: Results on the Full Whirlpool Compression Function. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 126–143. Springer, 2009. 1, 2.2
8. Meltem Sönmez Turan and Ray Perlner and Lawrence E . Bassham and William Burr and Donghoon Chang and Shu-jen Chang and Morris J. Dworkin and John M. Kelsey and Souradyuti Paul and Rene Peralta. Status Report on the Second Round of the SHA-3 Cryptographic Hash Algorithm Competition. Interagency Report 7764, National Institute of Standards and Technology, 2011. 1
9. Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer. Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher. In Michael J. Jacobson Jr. and Vincent Rijmen and Reihaneh Safavi-Naini, editor, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 2009. 1
10. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Orr Dunkelman, editor, *Fast Software Encryption*, volume 5665 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2009. 1, 2.2
11. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Rebound Attacks on the Reduced Grøstl Hash Function. In Josef Pieprzyk, editor, *The Cryptographers' Track at the RSA Conference(CT-RSA)*, volume 5985 of *Lecture Notes in Computer Science*, pages 350–365. Springer, 2010. 1, 1, 2.2
12. Ralph Charles Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, 1979. 1
13. NIST. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family, November 2007. This notice by NIST is available at `http://www.csrc.nist.gov/pki/HashWorkshop/timeline.html` with the Docket No: 070911510-7512-01. (Accessed on 30/10/2011). 1
14. NIST. Announcing the Development of New Hash Algorithms for the Revision of Federal Information Processing Standard (FIPS) 180-2, Secure Hash Standard, January 2007. This notice by NIST is available at `http://www.csrc.nist.gov/pki/HashWorkshop/timeline.html` with the Docket No: 061213336-6336-01. (Accessed on 30/10/2011). 1
15. NIST. Third (Final) Round Candidates. Official notification from NIST, 2009. Available at `http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html` (Accessed on 29/10/2011). 1
16. Thomas Peyrin. Improved Differential Attacks for ECHO and Grøstl. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 370–392. Springer, 2010. 1, 1, 2.3
17. Bart Preneel. Cryptographic Hash Functions: Theory and Practice. Invited Talk at INDOCRYPT 2010, 2010. 1

18. Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. Non-full-active Super-Sbox Analysis: Applications to ECHO and Grøstl. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 38–55. Springer, 2010. 1

A Security bounds for Grøstl-0 compression function against (K-target) preimage attacks

In Theorem 1 we prove two results for Grøstl-0 compression function under the assumption that the permutations P and Q are ideal and our results are also applicable to Grøstl compression function. The first result of Theorem 1 shows that for an adversary who makes q queries to the permutations, the success probability of a preimage attack on l-bit Grøstl-0 compression function is upper bounded by $\frac{(q^2+q)}{2^l}$ where $q^2+q \leq 2^l$. This proof is a more accurate compared to the one in [2] for the reasons mentioned in Section 3 and the results of both proofs are almost similar. The second result of Theorem 1 shows that the success probability of a K-target preimage attack on Grøstl-0 compression function after q queries is upper bounded by $K.((q^2+q)/2^l)$, where $q^2+q \leq 2^l$ by assuming that the permutations P and Q are ideal. Therefore, an adversary must make at least $2^{l/2}/\sqrt{K}$ queries to find a preimage of one of the K targets.

Theorem 1. *Consider an l-bit Grøstl-0 compression function based on ideal permutations P and Q. For a computationally unbounded adversary $\mathcal{A}$ making at most q queries to the permutations P and Q and their inverses, the advantage to find a preimage for Grøstl-0 compression function is upper bounded by $(q^2+q)/2^l$ where $q^2+q \leq 2^l$. When $\mathcal{A}$ is supplied with K-target hash values, the advantage to find a preimage for one of these hash values is upper bounded by $K.((q^2+q)/2^l)$ where $q^2+q \leq 2^l$.*

Proof:

Let x be the value whose preimage we desire for Grøstl-0 compression function, i.e., we want to find α, α' such that $x \oplus P(\alpha) \oplus \alpha \oplus Q(\alpha') \oplus \alpha' = 0$.

Suppose that $\mathcal{A}$ has made:

1. A total of i_1 queries to P, P^{-1} oracles and stored the input/output pairs $(\alpha_j, \beta_j = P(\alpha_j))$ (for $j = 1, ..., i_1$) for these i_1 queries in a list L_1.
2. A total of i_2 queries to Q, Q^{-1} oracles and stored the input/output pairs $(\alpha'_j, \beta'_j = Q(\alpha'_j))$ (for $j = 1, ..., i_2$) for these i_2 queries in a list L_2.

Now we claim that the probability that the next query of $\mathcal{A}$ will result in an answer that leads to a preimage with at most probability of:

$$p_{next} \leq \begin{cases} i_2/(2^l - i_1) \text{ if the next query is to } P \text{ or } P^{-1} \\ i_1/(2^l - i_2) \text{ if the next query is to } Q \text{ or } Q^{-1} \end{cases}$$

The justification for the first bound is as follows: Suppose that $\mathcal{A}$'s next query is to P at the value α, where α is not equal to one of the α_j's already in the list L_1. The value returned $\beta = P(\alpha)$ is chosen uniformly at random from the set of all $2^l - i_1$ l-bit strings which are not equal to one of the β_j values in L_1. In order to give a preimage, the β must be such that for some $j \in \{1, ..., i_2\}$,

$$x \oplus \beta \oplus \alpha \oplus Q(\alpha'_j) \oplus \alpha'_j = 0 \tag{2}$$

However, for each value of j, there is at most one value of β such that Eq.(2). is satisfied. Therefore, there are at most i_2 "good" values of β that will give a preimage, and since β is chosen uniformly among $2^l - i_1$ possible values, the probability that β will be "good" is at most $i_2/(2^l - i_1)$, as in the bound above. A similar argument applies to P^{-1} queries and a symmetric argument applies for queries made to Q or Q^{-1} giving the bound $i_1/(2^l - i_2)$.

Therefore, we conclude that if $\mathcal{A}$ makes $i_1 + i_2 = q - 1$ queries in total to all oracles so far, then its next q^{th} query will provide a preimage with probability $p_q \leq q/(2^l - q)$, since both bounds for p_{next} above are below this value. Therefore, the probability of a preimage being found in any of q queries is at most

$$p_{preimage} \leq \sum_{i=1,\ldots,q} p_q \leq \sum_{j=1,\ldots,q} j/(2^l - j) \leq q^2/(2^l - q) \leq q^2 + q/2^l \tag{3}$$

if $q^2 + q \leq 2^l$. This result is almost the same as the one claimed in [2].

Now assume that $\mathcal{A}$ is challenged with $K = 2^{s_K}$-target hash values, of which it needs to find a preimage for one of them. Equivalently, $\mathcal{A}$ has to find a preimage to a subset of $(l - s_K)$ output bits of the compression function. Therefore, Equation (2) can be modified to only look at the equality on these $l - s_K$ output bits. Let that l-bit state of Grøstl-0 is viewed as a byte oriented $b \times b$ matrix, that is $l = b \times b \times 8$ with matrix indices $1, \ldots c$ where $c = b \times b$. Now Equation (2) can be modified as follows: Let Z denote the subset of $\{1, \ldots, c\}$ corresponding to the indices of $(l - s_K)/8$ bytes in the $b \times b$ matrix, i.e the size of Z is $(l - s_K)/8$ bytes. For an $b \times b$ byte variable x, let $(x)_Z$ denote the restriction of x to the bytes in indices contained in Z. Then the modified Equation (2) is:

$$(x)_Z \oplus (\beta)_Z \oplus (\alpha)_Z \oplus (Q(\alpha'_j))_Z \oplus (\alpha'_j)_Z = 0 \tag{4}$$

This equation determines $(\beta)_Z$ uniquely, once x, α, α'_k and $Q(\alpha'_k)$ are determined. Since the values of α and β are still l bits long, for each value of $j = 1, ..., i_2$, there now at most $K = 2^{s_K}$ (instead of at most one) "good" values of β that satisfy Equation 4, hence overall at most $i_2.K$ for all values of j. Therefore, the bounds for p_{next}, and the final result in Equation 3 gets multiplied by K. Therefore, the advantage of $\mathcal{A}$ to find a preimage for one among $K = 2^{s_K}$ target hash values is at most $K.(q^2 + q)/2^l$. $\square$

Remark 1. The security bound derived for Grøstl-0 compression function against K-target preimage attacks in Theorem 1 does not depend on the details of P and Q permutations. Hence, it is reasonable to use this bound also for the generic security of the construction against chosen-K-target preimage attacks. Therefore, the complexities of chosen-K-target preimage attacks on Grøstl-0 compression function discussed in Section 4 are compared with respect to the generic bound of K-target preimage attacks derived in Theorem 1.

B Security bounds for Grøstl-0 hash function against K^*-target preimage attacks

In Theorem 2 we prove that finding a preimage for one of K^* hash values of Grøstl-0 hash function built from K targets of the $2n$-bit compression function has a probability less than $\frac{q.2^n \cdot K}{2^{2n} - (K+s-1)} + \frac{(q^2+q) \cdot K}{2^{2n}}$ where q is the number of queries and $q^2 + q \leq 2^{2n}$. The probability $\frac{q.2^n \cdot K}{2^{2n} - (K+s-1)}$ is approximately upper bounded by $q.K/2^n$ when $K + q$ is much smaller than 2^{2n}.

Therefore, the total success probability is less than $\frac{q \cdot K}{2^n} + \frac{(q^2+q)\cdot K}{2^{2n}}$. Since the first term dominates the second term, the adversary has to make at least $2^n/K$ queries to find a preimage of one of the K^* target hash values. Since permutations are assumed ideal, this analysis is also applicable to Grøstl hash function.

Theorem 2. *Consider an n-bit Grøstl-0 hash function with the compression function based on ideal permutations P and Q, and output transformation OT as $OT(x)= \mathrm{trunc}_n(P(x) \oplus x)$. If Y is a set of the compression function outputs selected independent of P and Q, and Y^* is the target hash value set such that $Y^* = \{OT(x) : x \in Y\}$; the probability of finding a preimage of one of the elements of Y^* is upper bounded by $\frac{q.2^n \cdot K}{2^{2n}-(K+s-1)} + \frac{(q^2+q)\cdot K}{2^{2n}}$, assuming that $q^2 + q \leq 2^{2n}$ (Whereas K^* and K are respectively the number of elements of the sets Y^* and Y).*

Proof:

Let x be the preimage for one of the hash targets in set Y^*. We can split the results into two independent cases as follows:

– Case I: x is the preimage of one of the targets in set Y^* such that $cf(x) \in Y$.
– Case II: x is the preimage of one of the targets in set Y^* such that $cf(x) \notin Y$.

The success probability of finding x as the preimage of one of the targets in Y^* is $P(\mathrm{I}) + P(\mathrm{II})$. We know from Theorem 1 that $P(\mathrm{I}) \leq (q^2 + q) \cdot K/2^{2n}$. Case II implies that there exists a query number s to P or to P^{-1}, such that, if $(z_s^*, P(z_s^*))$ denotes the input, output of P for the s'th query, and $z_1, ..., z_K$ denote the elements of Y, then there exists $i \in \{1, ..., K\}$ such that:

$$z_s^* \notin Y \text{ and } Trunc_n(P(z_s^*) \oplus z_s^*) = Trunc_n(P(z_i) \oplus z_i) \tag{5}$$

Let's consider the case that the s'th query is a query to P (Note that the case when it is a query to P^{-1} is similar). Given any fixed values for z_s^*, z_i and $P(z_i)$, there are 2^n values of $P(z_s^*) \in \{0,1\}^{2n}$ that satisfy Equation (5). Since there are K possible values for i, there are at most $2^n \cdot K$ "bad" values of $P(z_s^*)$ that satisfy Equation (5) for some i. We call this set of "bad" $P(z_s^*)$ values Bad_s. Now, conditioned on the already fixed values of $P(z_r)$ for $r = 1, ..., K$ and $P(z_r^*)$ for $r = 1, ..., s-1$, the value $P(z_s^*)$ returned by as answer to the adversary's s'th query z_s^* to P is uniformly random in a set of size $2^{2n} - (K + s - 1)$. Therefore, the probability that $P(z_s^*) \in Bad_s$ is at most $p_s = \frac{|Bad_s|}{2^{2n}-(K+s-1)} \leq \frac{2^n \cdot K}{2^{2n}-(K+s-1)}$. Taking a union over all possible $s = 1, ..., q$, we obtain that $P(\mathrm{II}) \leq \sum_{s=1,...,q} p_s \leq \frac{q \cdot 2^n \cdot K}{2^{2n}-(K+q)}$. Hence, the success probability of finding preimage of one of the elements of Y^* is $P(\mathrm{I}) + P(\mathrm{II})$ which is upper bounded by $\frac{q.2^n \cdot K}{2^{2n}-(K+s-1)} + \frac{(q^2+q)\cdot K}{2^{2n}}$ where $q^2 + q \leq 2^{2n}$.□

Remark 2. In Theorem 2 we applied a more general approach to derive a security bound for finding a preimage of one of K^* hash function targets that are computed from K compression function targets instead of specifically deriving a security bound in the chosen-K^*-target preimage attack setting. Recall that in the chosen-K^*-target preimage attacks on the hash function, K^* targets partly depend on P and Q and computed from K-target compression function values. Since the set Y of K compression function outputs are selected independent of P and Q, the security bound derived in Theorem 2 is also applicable to chosen-K^*-target preimage attack on the hash function where K^* targets are determined by the K targets of the compression function. As a planned addition in the full version, we consider proving a more accurate generic complexity with ideal P and Q for the attacks where the chosen targets partially depend on P and Q.

C Additional truncated differential trails

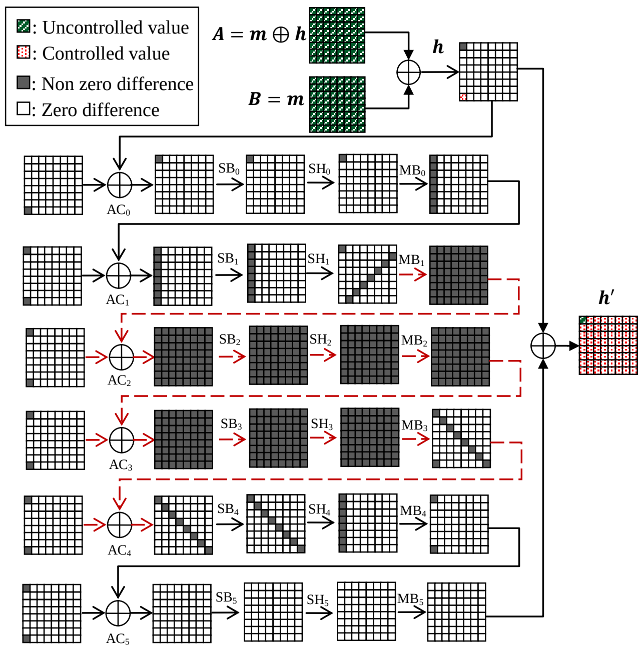
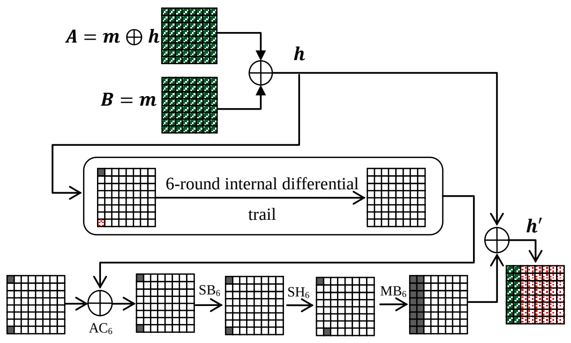
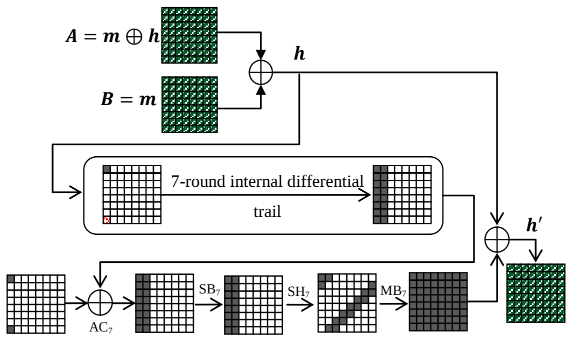


Figure 7: Internal differential trail for 2^8-target pseudo preimage attack on 6-round Grøstl-0



(a) 7-round internal differential trail for 2^{24} targets



(b) 8-round internal differential trail for 2^{136} targets

Figure 8: Extension of the internal differential trail of Figure 7 for chosen-K-target preimage attacks

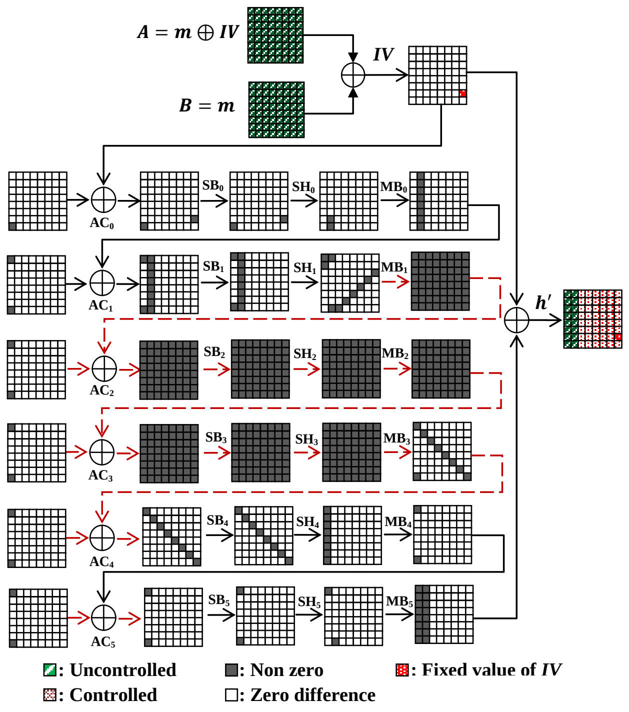


Figure 9: Internal differential trail for 2^{16}-target preimage attack on 6-round 256-bit Grøstl-0

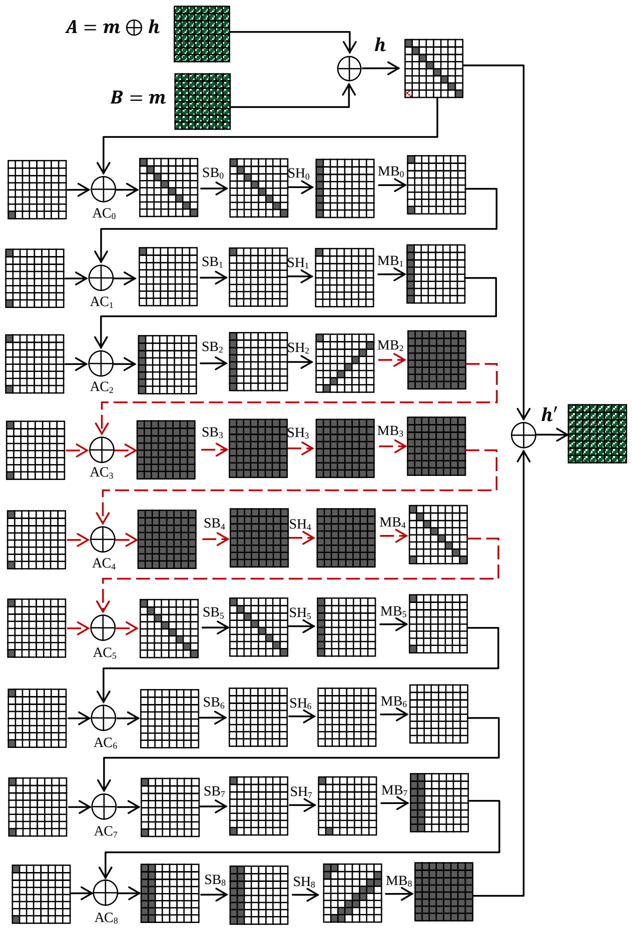


Figure 10: Internal differential trail for 2^{192}-target preimage attack on 9-round 256-bit Grøstl-0