Improved Collision Attacks on the Reduced-Round Grøstl Hash Function

Kota Ideguchi[1,3], Elmar Tischhauser[1,2,⋆], and Bart Preneel[1,2]

[1] Katholieke Universiteit Leuven, ESAT-COSIC and [2] IBBT
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
[3] Systems Development Laboratory, Hitachi, LTD.
Yoshida-cho 292, Totsuka, Yokohama, Kanagawa, Japan
kota.ideguchi.yf@hitachi.com
{elmar.tischhauser,bart.preneel}@esat.kuleuven.be

Abstract. We analyze the Grøstl hash function, which is a 2nd-round candidate of the SHA-3 competition. Using the start-from-the-middle variant of the rebound technique, we show collision attacks on the Grøstl-256 hash function reduced to 5 and 6 out of 10 rounds with time complexities 2^{48} and 2^{112}, respectively. Furthermore, we demonstrate semi-free-start collision attacks on the Grøstl-224 and -256 hash functions reduced to 7 rounds and the Grøstl-224 and -256 compression functions reduced to 8 rounds. Our attacks are based on differential paths between the two permutations P and Q of Grøstl, a strategy introduced by Peyrin [15] to construct distinguishers for the compression function. In this paper, we extend this approach to construct collision and semi-free-start collision attacks for both the hash and the compression function. Finally, we present improved distinguishers for reduced-round versions of the Grøstl-224 and -256 permutations.

Key words: Hash Function, Differential Cryptanalysis, SHA-3

1 Introduction

Cryptographic hash functions are one of the most important primitives in cryptography; they are used in many applications, including digital signature schemes, MAC algorithms, and PRNG. Since the discovery of the collision attacks on NIST's standard hash function SHA-1 [17, 4], there is a strong need for a secure and efficient hash function. In 2008, NIST launched the SHA-3 competition [14], that intends to select a new standard hash function in 2012. More than 60 hash functions were submitted to the competition, and 14 functions among them were selected as second-round candidates in 2009. The hash functions are expected to be collision resistant: it is expected that finding a collision takes $2^{n/2}$ operations for a hash function with an n-bit output.

Several types of hash functions have been proposed to the SHA-3 competition. Some of the most promising designs reuse the components of the AES [1]. In this paper, we concentrate on the Grøstl [5] hash function that belongs to this class.

To analyze AES-type functions, several cryptanalytic techniques were developed. Among them, one of the most important techniques is truncated differential cryptanalysis [7, 16]. In this technique, differences of bits are reduced to differences of bytes. This

⋆ F.W.O. Research Assistant, Fund for Scientific Research – Flanders.

makes differential paths very simple and is very suitable for functions based on byte-wise calculations, such as AES-type functions. Refinements of truncated differential cryptanalysis are the rebound attack [11] and the two variants thereof: the start-from-the-middle and the Super-Sbox rebound attacks [12, 9, 6]. As an AES-based design, the resistance of Grøstl against these attacks has been assessed in detail [11–13, 6, 15]. Specifically, as a comparison with our results, a collision attack on the Grøstl-224 and -256 hash functions reduced to 4 rounds with 2^{64} time and memory complexities was shown by Mendel *et al.* [13], and a semi-free-start collision attack on the Grøstl-256 compression function reduced to 7 rounds with 2^{120} time and 2^{64} memory complexities is shown by Mendel *et al.* [13] and Gilbert *et al.* [6].

We use the start-from-the-middle and the Super-Sbox variants of the rebound technique to analyze the reduced-round Grøstl-224 and Grøstl-256 hash functions, compression function, and permutations. We show collision attacks on the Grøstl-256 hash function reduced to 5 and 6 rounds with time complexities 2^{48} and 2^{112}, respectively, and on the Grøstl-224 hash function reduced to 5 rounds with time complexity 2^{48}. To the best of our knowledge, these are the best collision attacks on Grøstl-224 and 256 hash functions so far. Our analysis uses truncated differential paths between the two permutations, P and Q. This approach was introduced in [15] to construct distinguishers. Generalizing our collision attacks to semi-free-start collision attacks, we construct semi-free-start collision attacks on the Grøstl-224 and -256 hash functions reduced to 7 rounds with time complexity 2^{80} and on the Grøstl-224 and -256 compression functions reduced to 8 rounds with time complexity 2^{192}. To the best of our knowledge, these are the best semi-free-start collision attacks on the Grøstl-224 and -256 hash functions and compression functions so far. By contrast, the paths in [15] could not be used to obtain semi-free-start collisions due to lack of degrees of freedom. Furthermore, we show distinguishers of the Grøstl-224 and -256 permutations reduced to 7 and 8 rounds. The 7-round distinguisher has a practical complexity. We list an example of input pairs of the permutations.

Our cryptanalytic results of Grøstl are summarized in Table 1. We will explain these results in Sect. 4 and 5.

Table 1. Summary of results for Grøstl

Target	Hash Length	Rounds	Time	Memory	Type	Reference
hash function (full: 10 round)	224, 256	4	2^{64}	2^{64}	collision	[13]
	256	5	2^{48}	2^{32}	collision	Sect. 4.2
	256	6	2^{112}	2^{32}	collision	Sect. 4.2
	224	5	2^{48}	2^{32}	collision	Sect. 4.2
	224, 256	7	2^{80}	2^{64}	semi-free-start collision	Sect. 4.2
compression function (512-bit CV)	256	7	2^{120}	2^{64}	semi-free-start collision	[13, 6]
	224, 256	8	2^{192}	2^{64}	semi-free-start collision	Sect. 4.3
permutation	224, 256	7	2^{55}	-	distinguisher	[12]
	224, 256	7	2^{19}	-	distinguisher	Sect. 5.2
	256	8	2^{112}	2^{64}	distinguisher	[6]
	224, 256	8	2^{64}	2^{64}	distinguisher	Sect. 5.1

The paper is organized as follows. In Sect. 2, the specification of Grøstl is briefly explained. In Sect. 3, we give a short review of the techniques used in our analysis. In Sect. 4 and 5, we analyze Grøstl. Sect. 6 concludes the paper.

2 A Short Description of Grøstl

Here, we shortly explain the specification of Grøstl. For a detailed explanation, we refer to the original paper [5]. Grøstl is an iterated hash function with an SPN structure following the AES design strategy. The chaining values are 512-bit for Grøstl-224 and -256; they are stored as an 8 by 8 matrix whose elements are bytes. The compression function takes a 512-bit chaining value (CV) and a 512-bit message block as inputs and generates a new 512-bit CV. The compression function uses two permutations P and Q, with input length 512-bit. The compression function is computed as follows (see also Fig. 1):

$$x = \mathrm{Comp}_{\mathrm{Grøstl}}(y, z) = P(y \oplus z) \oplus Q(z) \oplus y,$$

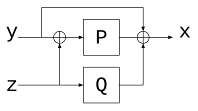

Fig. 1. The Compression Function of Grøstl

A message is padded and divided into 512-bit message blocks $m_1, \ldots, m_b$ and processed as follows to generate a hash value h,

$$\begin{aligned} CV_0 &= IV, \\ CV_i &= \mathrm{Comp}_{\mathrm{Grøstl}}(CV_{i-1}, m_i), \quad i = 1, \ldots, b, \\ h &= \mathrm{Output}(CV_b) = P(CV_b) \oplus CV_b. \end{aligned}$$

Here, the IV is a constant value, namely the binary representation of the hash length. For example, the IV of Grøstl-256 is $IV = \texttt{0x00}\ldots\texttt{0100}$; its matrix representation is given in Fig. 2(a).

The permutation P is an AES-type SP-network structure [1] with 10 rounds, where a round consists of `MixBytes` ∘ `ShiftBytes` ∘ `SubBytes` ∘ `AddConstants`. The permutation Q has the same structure; it uses the same `MixBytes`, `ShiftBytes` and `SubBytes` but another `AddConstants`.

`SubBytes` is a non-linear transformation using the AES S-box. `ShiftBytes` consists of byte-wise cyclic shifts of rows. The i-th row is moved left cyclically by i bytes. `MixBytes` is a matrix multiplication, where a constant MDS matrix is multiplied from the left.

`AddConstants` of P xors the round number to the first bytes of the internal state. `AddConstants` of Q xors the negation of the round number to the eighth byte of the internal state. These functions are depicted in Fig. 2(b).

3 Cryptanalytic Techniques

Here, we shortly explain the start-from-the-middle and Super-Sbox variants of the rebound technique. In this paper, we call these the start-from-the-middle rebound technique

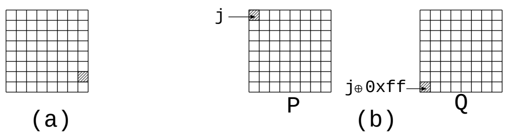

Fig. 2. (a) The IV of Grøstl-256 (b) `AddConstants` at the j-th round of the Grøstl permutations

and the Super-Sbox rebound technique, respectively. For a detailed explanation, we refer to the original papers [12, 9, 6].

3.1 The Start-From-the-Middle Rebound Technique

The start-from-the-middle rebound technique was introduced by Mendel *et al.* [12] and can be considered as a generalization of the rebound attack [11]. We explain this technique using a small example with a 4-round path depicted in Fig. 3.

We split the path into a controlled phase and an uncontrolled phase. The controlled phase consists of the middle part of the path and comprises most costly differential transitions. In Fig. 3, the controlled phase is shown by solid arrows and the uncontrolled one is shown by dashed arrows.

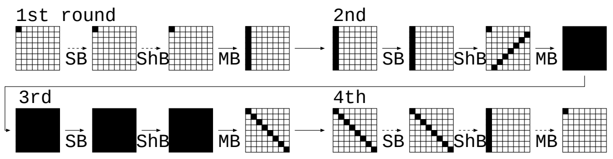

Fig. 3. Example path for the start-from-the-middle rebound technique

At first, the adversary builds $2^8 - 1$ difference distribution tables (DDT) of the Sbox; each DDT is a table that lists for a given output difference the input differences and the corresponding input pairs. There are 2^8 tables of size 2^8. Then, the adversary deals with the controlled phase. The procedure for the controlled phase consists of the following three steps.

1. The adversary fixes the input difference of the 4th round `SubBytes`, then calculates the output difference of the 3rd round `SubBytes`.
2. He fixes the input difference of the 2nd round `MixBytes` and calculates the input difference of the 3rd round `SubBytes`. Then, for each column of the output of the 3rd round `SubBytes`, he finds pairs of values of the column by using the precomputed difference distribution table such that the fixed input and output differences of the 3rd-round `SubBytes` are followed. This can be done for each column independently. By repeating this step with different input differences of the 2nd-round `MixBytes`, he obtains a set of solutions for each column, hence eight sets of solutions.
3. For the solutions obtained in the previous step, he calculates the differences of the input of the 2nd-round `SubBytes`. Then, he looks for solutions such that the difference of the input of the 2nd-round `SubBytes` is transformed to a one-byte difference by the 1st-round inverse `MixBytes`. These are the solutions of the controlled phase. He

can repeat the procedure from Step. 1 to 3 by changing the input difference of the 4th round `SubBytes`.

Although in the above procedure the adversary proceeds backwards from the end of the 3rd round to the input of the 1st round `MixBytes`, he can also proceed in the opposite direction. In this procedure, the average time complexity to find one solution of the controlled phase is one and the memory complexity is negligible.

The uncontrolled phase is probabilistic. In the backward direction (from the input of the 1st-round `ShiftBytes` to the beginning of the path), the probability to follow the path is almost one. In the forward direction (from the input of the 4th-round `SubBytes` to the end of the path), it requires 2^{56} solutions of the controlled phase in order to follow a $8 \rightarrow 1$ transition for the 4th-round `MixBytes`.

In total, generating one solution of the whole path takes time 2^{56}.

The degrees of freedom can be counted by the method of [15]. For the path above, we can find about 2^{15} solutions following the whole path. Note that if we consider differential paths between $P(m+IV)$ and $Q(m)$, we need not to halve the overall degrees of freedom at the middle point. Hence for this case, the degrees of freedom are doubled compared with the number counted by the method of [15].

3.2 The Super-Sbox Rebound Technique

The Super-Sbox rebound technique [9, 6] is a composition of the super box concept and the rebound technique. The super box concept [2, 3] considers the Sbox layers of two consecutive rounds as one big Sbox layer. Specifically, by exchanging the order of the first `SubBytes` and the first `ShiftBytes`, we have a composition of `SubBytes` ∘ `AddConstants` ∘ `MixBytes` ∘ `SubBytes`. This composition is considered as eight independent 64-bit to 64-bit Sboxes, which are called super boxes, or Super-Sboxes as in [6].

We explain the Super-Sbox Rebound technique in an example in Fig. 4. Similar to the start-from-the-middle rebound technique, we split the path into a controlled phase and an uncontrolled phase.

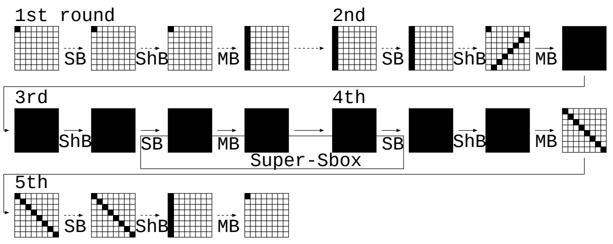

Fig. 4. Example path for the Super-Sbox rebound technique

For the controlled phase, the procedure proceeds as follows.

1. The adversary fixes the input difference of the 2nd round `MixBytes`, then calculates the input difference of the 3rd round `SubBytes`. For each Super-Sbox, he computes the output differences of the Super-Sbox for all possible pairs of inputs which have the

fixed input difference and makes tables of the output difference and the corresponding input pair. This takes time and memory 2^{64}.
2. Then, he fixes the output difference of the 4th round `Mixbytes` and calculates the output difference of the 4th round `SubBytes`.
3. For each Super-Sbox, he looks for the pairs of the inputs of the Super-Sbox by using the difference distribution table of Step 1 such that the output difference of the 4th-round `SubBytes` is equal to that of Step 2. This can be done for each Super-Sbox independently. If the number of solutions of the controlled phase is not sufficient, he can repeat Step 2 and 3 until the output differences of the 4th round `MixBytes` are exhausted. After exhausting the differences at Step 2, he can repeat the procedure from Step 1 by changing the input difference of the 2nd round `MixBytes`.

Although in the above procedure the adversary proceeds forward from the input of the 2nd round `MixBytes` to the end of the 4th round, he can also proceed in the opposite direction.

As before, the uncontrolled phase is probabilistic. Since there are two $8 \to 1$ transitions (the 1st-round `MixBytes` and the 5th-round `MixBytes`), he needs $2^{56\times 2}$ solutions of the controlled phase. Hence, the time complexity to generate a solution following the whole path is 2^{112}. The total memory complexity is 2^{64}.

The degrees of freedom can be counted as in the case of the start-from-the-middle rebound attack. For the path above, we can find about 2^{15} solutions following the whole path.

This path can also be followed by using the start-from-the-middle rebound technique, requiring the same time and memory complexity as the Super-Sbox rebound technique.

4 Collision and Semi-Free-Start Collision Attacks on Reduced Round Grøstl

4.1 The Attack Strategy

We show collision attacks and semi-free-start collision attacks on the reduced-round Grøstl hash function and compression function. We apply the start-from-the-middle rebound technique on the 5-round, 6-round and 7-round Grøstl hash functions and the Super-Sbox rebound technique on the 8-round Grøstl compression function.

In all of these cases, we consider differential paths between $P(m \oplus IV)$ and $Q(m)$. (For collision attacks, the IV is the value specified by the designer. For semi-free-start collision attacks, the IV can take another value.) Since the permutations P and Q only differ in the `AddConstants` functions, where there are two-byte differences per round, we can construct differential paths between P and Q which hold with high probability. This strategy was introduced in [15] to construct distinguishers of the compression function.

In our attacks on the hash function, an adversary finds a message pair in which each padded message has the same number b of blocks with $b \geq 2$. Consider the case $b = 2$. Firstly, he finds a pair of the first blocks m_1 and m'_1 that generate an internal collision of CV after processing the first block:

$$CV = \mathrm{Comp}_{\mathrm{Grøstl}}(IV, m_1) = \mathrm{Comp}_{\mathrm{Grøstl}}(IV, m'_1)\,. \tag{1}$$

Then, the adversary appends the same message block m_2 to both blocks such that the padding rule is satisfied;

$$\text{Padding}(M) = m_1||m_2, \quad \text{Padding}(M') = m'_1||m_2\,, \tag{2}$$

where M and M' are messages which generate the same hash value,

$$\text{hash}(M) = \text{hash}(M') = \text{Output}(\text{Comp}_{\text{Grøstl}}(CV, m_2))\,. \tag{3}$$

As finding the second message blocks is easy, we can concentrate on finding the first blocks.

For an attack on the compression function, finding the first blocks is sufficient.

4.2 Applying the Start-from-the-Middle rebound technique to 5-round, 6-round, and 7-round Grøstl-224 and -256

We show collision attacks on the Grøstl-256 hash function reduced to 5 rounds and 6 rounds and a semi-free-start collision attack on the Grøstl-224 and -256 hash functions reduced to 7 rounds. The full Grøstl-224 and -256 hash function has 10 rounds.

Collision attack on the Grøstl-256 hash function reduced to 5 rounds First, we explain a collision attack on the Grøstl-256 hash function reduced to 5 rounds. For this case, our differential path between $P(m \oplus IV)$ and $Q(m)$ of the first message block is shown in Fig 5. The controlled phase is shown by solid arrows and the uncontrolled phase by dashed arrows.

The controlled phase starts from the input of the 4th round `SubBytes`, proceeds backwards and ends at the input of the 1st round `MixBytes`. The average time complexity to generate an internal state pair which follows the path of the controlled phase is one. The remaining steps of the path are the uncontrolled phase. For the forward direction (from the 4th round `SubBytes` to the end of compression function), the probability to follow the path is almost one. For the backward direction (from the 1st round inverse `ShiftBytes` to the beginning of the compression function), it takes 2^{16} computations to follow the inverse `AddConstants` and addition of the IV in the 1st round. Therefore, the time complexity to find a message block to follow the whole path is 2^{16}. An example of such a message block can be found in Table 2.

As the differences of CV at the end of the path are determined by the 8-byte difference before the final `MixBytes`, by the birthday paradox we need to have 2^{32} message blocks following the path in order to obtain a pair (m_1, m'_1) whose CVs collide; $P(m_1 \oplus IV) \oplus Q(m_1) \oplus IV = P(m'_1 \oplus IV) \oplus Q(m'_1) \oplus IV$. Therefore, the total complexity of the attack is $2^{16+32} = 2^{48}$ time and 2^{32} memory.

Table 2. Message block m following the differential path between P and Q used in the 5-round collision attack.

m	7d108fbb55b235a4	0eba3e293ec86701	42e5de7469e6e097	cbaf6719a603b7bf
	b617d48098448877	215829419c0a161c	01131ad85ba62da2	b9dbe9d6fb8e44ce

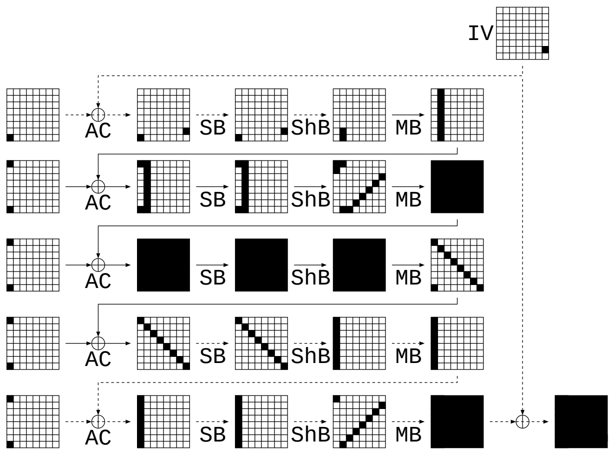

Fig. 5. Differential path between P and Q of 5-round Grøstl-256

By counting the degrees of freedom, it is expected that we can generate at most 2^{64} message blocks following the path. Because the attack requires 2^{32} message blocks following the path, we have enough degrees of freedom to make the attack work.

The 5-round path can be modified to be used in a semi-free start collision attack on the Grøstl-224 and -256 hash functions reduced to 7 rounds. The total complexity of the attack is 2^{80} time and 2^{64} memory. The 7-round path is depicted in Fig 10 in Appendix A. In this case, we can generate at most 2^{64} message blocks following the path. On the other hand, the attack needs 2^{64} message blocks following the path to generate a collision of CV, because CV is determined by 8 non-zero bytes of the IV and the 8-byte difference before the last `MixBytes`. Hence, it is expected that we can generate just one collision of CV with high probability.[1]

The 5-round path can also be slightly modified to be used in a collision attack on the Grøstl-224 hash function reduced to 5 rounds. The path is depicted in Fig. 11 in Appendix A. Note that the IV of Grøstl-224 has a non-zero byte only at the least significant byte.

Collision attack on the Grøstl-256 hash function reduced to 6 rounds Next, we show a collision attack on the 6-round Grøstl-256 hash function. The path used is depicted in Fig. 6. The controlled phase (solid line) starts from the input of the 2nd round `MixBytes`, proceeds forwards and ends at the input of the 5th round `SubBytes`. The average time complexity to generate an internal state pair which follows the path of

[1] If we cannot find a collision with this path, we can change the path, such that there are a full active 'diagonal column' and a full active 'off-diagonal column' at the output of the 5th round `MixBytes`, instead of a full active diagonal column. (A diagonal column denotes diagonal 8 bytes, which are aligned to the first column by `ShiftBytes`, and a off-diagonal column denotes 8 bytes which are aligned to a column other than the first column by `ShiftBytes`.) For the new path, we can generate at most 2^{128} message blocks following the path. This is enough to make a collision of CV, because CVs live in 192-bit space. For this case, the complexities of the attack are 2^{112} for time and 2^{96} for memory.

the controlled phase is one. The remaining steps of the path are the uncontrolled phase (dashed line). For the forward direction (from the 5th round `SubBytes` to the end of compression function), the probability to follow the path is almost one. For the backward direction (from the 2nd round inverse `ShiftBytes` to the beginning of the compression function), it takes 2^{16} computations to follow the 2nd round inverse `AddConstants`, 2^{48} computations for the 2nd round inverse `MixBytes`, and 2^{16} computation for the inverse `AddConstants` and addition of the IV in the 1st round. Therefore, the time complexity to find a message block which follows the whole path is 2^{80}.

One might think that finding solutions of the internal state pair at Step 3 of the controlled phase is difficult and requires much time and memory. Specifically, we can generate 2^{32} solutions of the controlled phase with time and memory 2^{32}, hence the average cost to find one solution is one. We explain a way to do this in the next paragraph.

Before starting Step 3, the adversary prepares 2^{32} solutions for each column at the output of the 3rd round `MixBytes` in Step 2 independently. Then, a procedure to obtain solutions of the controlled phase in Step 3 is as follows.

3-1 He evolves the solutions obtained in Step 2 forwards until the input of the 4th round `MixBytes`. Now, he has 2^{32} solutions for each anti-diagonal or anti-off-diagonal column (An anti-diagonal column denotes anti-diagonal 8 bytes, which are aligned to the eighth column by inverse `ShiftBytes`, and an anti-off-diagonal column denotes 8 bytes which are aligned to a column other than the eighth column by inverse `ShiftBytes`.) at the input of the 4th round `MixBytes`.

3-2 He picks up an element among the solutions corresponding to the anti-diagonal column. This determines 4 bytes of the anti-diagonal 8-byte difference at the input of the 4th round `MixBytes`. As the transition pattern of the 4th round `MixBytes` in the path imposes 28 bytes linear equations on the input difference, the other 28-byte difference at the input of the 4th round `MixBytes` is fixed by solving these 28 bytes equations. Then, he checks whether this 28-byte difference is included in the other sets of the solutions (corresponding to anti-off-diagonal columns). He repeats this procedure for all 2^{32} elements of the solutions corresponding to the anti-diagonal column.

This procedure takes 2^{32} time and 2^{32} memory. Because he has $2^{32\times 8}$ candidates of solution at the input of the 4th round `MixBytes`, he can obtain $2^{32\times 8}/2^{28\times 8} = 2^{32}$ solutions at the output of the 4th round `MixBytes` through this procedure. Hence, he can generate one solution of the controlled phase with average complexity one.

As in the case of the 5-round collision attack, we need to have 2^{32} message blocks following the path in order to obtain a pair (m_1, m_1') whose CVs collide. Therefore, the total complexity of the attack is $2^{80+32} = 2^{112}$ time and 2^{32} memory.

By counting the degrees of freedom, it is expected we can generate at most 2^{32} message blocks following the path. Because the attack requires 2^{32} message blocks following the path, we expect just one collision of CV with high probability. [2] Hence our attack works.

[2] If we cannot find a collision with this path, we can change the path slightly; we can change the location of the full active 'off-diagonal' columns at the output of the 3rd round `MixBytes`. Using these paths, we can generate more message-blocks which have the same truncated difference pattern at the end of the compression function and we can obtain a sufficient number of message blocks for the attack to work.

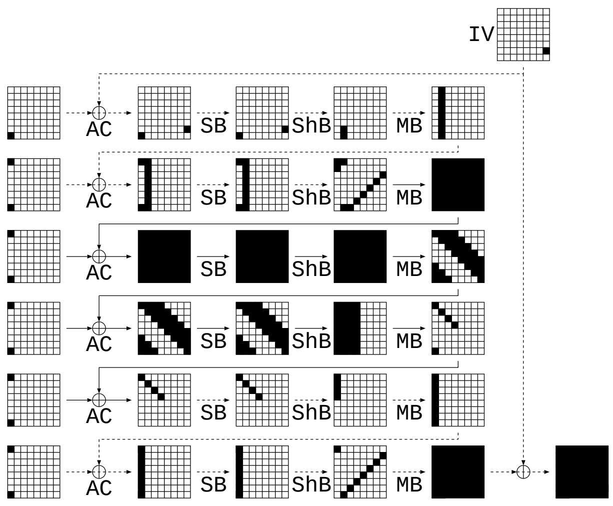

Fig. 6. Differential path between P and Q of 6-round Grøstl-256

4.3 Applying the Super-Sbox rebound technique to 8-round Grøstl-224 and -256

We show a semi-free-start collision attack on the Grøstl-224 and -256 compression functions reduced to 8 rounds, using the Super-Sbox rebound technique. The differential path between $P(m \oplus IV)$ and $Q(m)$ of the message block is shown in Fig 7.

The controlled phase (solid line) starts at the input of the 4th round `MixBytes` and ends at the input of the 7th round `SubBytes`. The average time complexity to generate an internal state pair that follows the path of the controlled phase is one. The remaining steps of the path are the uncontrolled phase (dashed line). For the forward direction (from the 7th round `SubBytes` to the end of compression function), the path can be followed with probability almost one. For the backward direction (from the 4th round inverse `ShiftBytes` to the beginning of the compression function), it takes 2^{112} computations to follow two $8 \rightarrow 1$ transitions at the 3rd round inverse `MixBytes` and 2^{16} computation to follow the 3rd round inverse `AddConstants`. Therefore, the time complexity to find a message block which follows the whole path is 2^{128}.

Because the differences of CV at the end of the path are determined by the 8-byte difference before the final `MixBytes` and the 8 non-zero bytes of IV, we need to have 2^{64} message blocks following the path in order to obtain a pair (m_1, m_1') whose CVs collide, $P(m_1 \oplus IV') \oplus Q(m_1) \oplus IV' = P(m_1' \oplus IV') \oplus Q(m_1') \oplus IV'$. Therefore, the total complexity of the attack is $2^{128+64} = 2^{192}$ for time and 2^{64} for memory.

By counting the degrees of freedom, it is expected that we can generate at most 2^{64} message blocks following the path. Because the attack requires 2^{64} message blocks following the path, it is expected for us to have just one collision of CV with high probability. [3] Hence our attack works.

[3] If we cannot find a collision with this path, we can change the path, such that there are a full active diagonal column and a off-diagonal column at the output of the 6th round `MixBytes`, instead of a full active diagonal column. For the new path, we can generate at most 2^{128}

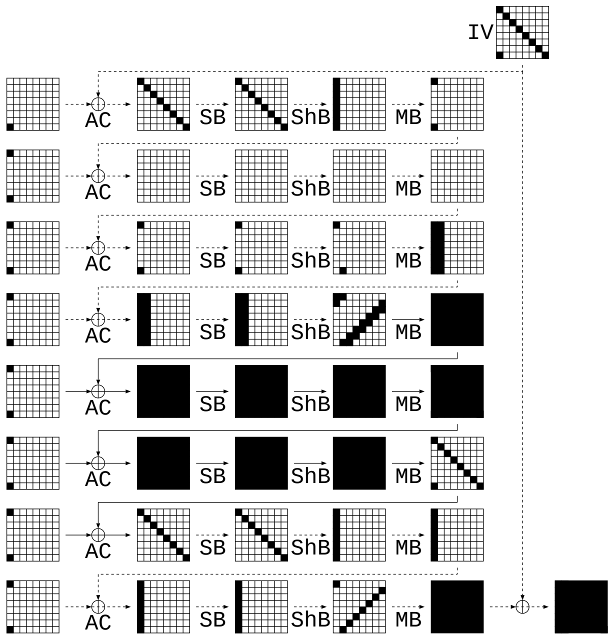

Fig. 7. Differential path between P and Q of 8-round Grøstl-224 and -256

We can follow the path also using the start-from-the-middle rebound technique with the same time complexity and the same memory complexity.

5 Distinguishers for the round-reduced Grøstl permutation

In this section, we show improved distinguishers for 7 and 8 rounds of the Grøstl permutations, applicable to both P and Q; the distinguisher for 7 rounds has practical time and memory requirements. The distinguishers work in both the limited-birthday [6] and the stronger Subspace Problem model [10]. Before going into detail about the different notions of distinguishing, we describe the differential paths and attack procedures.

The truncated differential path for Grøstl reduced to 7 rounds is depicted in Fig. 8.

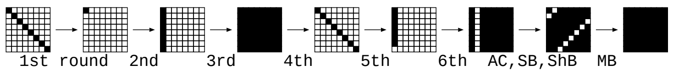

Fig. 8. Differential path for the 7-round distinguisher of the Grøstl permutation.

By using the start-from-the-middle rebound technique, we can create a pair following this path with a complexity of about 2^8 compression function calls and negligible memory. Starting from the input to `SubBytes` in the 5th round, we can create a pair following

message blocks following the path. This is enough to make a collision of CV, because CVs live in 192-bit space. For this case, the complexities are 2^{224} time and 2^{96} memory.

the path back to the output of `SubBytes` in the 2nd round with a complexity of about one. The remaining steps in both directions are uncontrolled. Except for the transition from 8 to 7 active bytes in the 5th round (which happens with probability 2^{-8}), they are followed with probability of almost one, hence about 2^8 repetitions are sufficient to generate one pair following the entire path.

For eight rounds, we use the truncated differential path illustrated in Fig. 9.

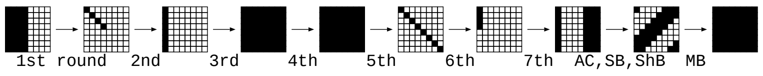

Fig. 9. Differential path for the 8-round distinguisher of the Grøstl permutation.

In order to afford the two fully active states in the middle, we employ the Super-Sbox technique. Starting from the output of `MixBytes` in round 3, 2^{64} pairs following the path until the input of `MixBytes` in round 6 can be generated at a cost of 2^{64} computations and 2^{64} memory. Out of these 2^{64} pairs, a fraction of about $2^{-2\cdot 32}$ are expected to follow the required $8 \rightarrow 4$ transitions in both backward and forward direction. The remaining transitions have a probability of about one, so that one pair following the entire 8-round path can be found with 2^{64} time and memory.

5.1 Distinguishers in the Limited-Birthday model

A limited-birthday distinguisher for a keyed permutation consists of an efficient procedure to obtain pairs of inputs and outputs such that the input and output differences are zero at i and j bit positions, respectively. If this procedure is more efficient than the conceived best generic algorithm based on the birthday paradox, it is considered a valid distinguisher [6]. Although primarily targeted for a known-key setting for keyed permutations, limited-birthday distinguishers have been applied to the Grøstl permutation and compression function [6].

In this setting, obtaining input/output pairs for the seven-round Grøstl permutation (barring the last `MixBytes`) having a zero difference at 448 input and 64 output bits should ideally take 2^{32} operations, while following our procedure has a complexity of 2^8. We have implemented the algorithm for the seven-round distinguisher. An example pair of inputs to the P permutation following the entire path can be found in Table 3.

Likewise, in the eight-round case, the ideal complexity is 2^{128}, while our procedure takes 2^{64} time and memory.

Table 3. Input pair (a, b) of Grøstl's P permutation following the differential path used in the 7-round distinguisher.

a	10becfbee55034d9	05598137bd731e89	d36c10b47aa2df07	5d81efd7fc3c893b
	d693175875f288e1	aed8001c310642cb	67cadc2fc955410a	e775ab2a9d6101f7
b	69becfbee55034d9	05538137bd731e89	d36c33b47aa2df07	5d81ef9ffc3c893b
	d693175898f288e1	aed8001c31b842cb	67cadc2fc955f50a	e775ab2a9d6101ce

5.2 Distinguishers in the Subspace Problem model

Distinguishers based on the Subspace Problem [10] consider the problem of obtaining t difference pairs for an N-bit permutation such that the output differences span a vector space of dimension less than or equal to n (provided the input differences span a vector space of dimension less than or equal to n too). Contrary to the limited-birthday model, lower bounds for the number of permutation queries needed in the generic case can be proven [10], so this provides a stronger distinguishing setting.

For the Grøstl permutation, we have $N = 512$. Our procedure to generate pairs for the seven round trail of Figure 8 has to be compared to the generic lower bound given by Corollary 4 of [10] with $n = 448$. Note that the conditions stated in Proposition 2 can actually be relaxed to $t \geq 2n$ and $N > n$. Starting from $t = 2^{11}$, our method is more efficient (2^{19} computations) than the generic algorithm (2^{23} queries), yielding a valid distinguisher.

For eight rounds, we have $n = 256$, and again choosing $t = 2^{11}$ gives a complexity of 2^{101} for the generic case, while our method has a complexity of 2^{75} time and 2^{64} memory.

6 Conclusion

We analyzed the Grøstl hash functions in this paper. Using the start-from-the-middle rebound technique, we have shown a collision attack on the Grøstl-256 hash function reduced to 5 rounds and 6 rounds with 2^{48} and 2^{112} time complexities, respectively. Furthermore, semi-free-start collision attacks on the Grøstl-224 and -256 hash functions reduced to 7 rounds, on the the Grøstl-224 and -256 compression functions reduced to 8 rounds were shown and some improvements of distinguishers of the permutations were presented. While our analysis shows (semi-free-start) collision attacks for up to seven rounds which leaves just three rounds of security margin, it does not pose a direct threat to the full Grøstl hash or compression function.

Further improvements. We expect memoryless algorithms for the birthday problem such as Floyd's cycle finding algorithm [8] to be applicable to the birthday matches required in the last phase of our collision attacks. In this case, the memory requirements of the collision attacks on 5 rounds and the semi-free-start collision attacks on 7 rounds of the Grøstl-224 and -256 hash functions become negligible without significant increase in time complexity. The memory requirements of the other attacks remain unchanged.

Acknowledgments We would like to thank Christian Rechberger for useful discussions and his suggestion for the eight-round distinguisher, and Kunihiko Miyazaki and Vincent Rijmen for their comments on the paper. The work in this paper was supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II and in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government.

References

1. J. Daemen and V. Rijmen, Design of Rijndael, Springer, 2001.

2. J. Daemen and V. Rijmen, Understanding Two-Round Differentials in AES, SCN, LNCS 4116, pp. 78-94, 2006.
3. J. Daemen and V. Rijmen, Plateau Characteristics, Information Security, IET, vol. 1-1, pp. 11-17, 2007.
4. C. De Cannière and C. Rechberger, "Finding SHA-1 Characteristics: General results and applications," ASIACRYPT 2006, LNCS 4284, pages 1-20, Springer, 2006.
5. P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen, Grøstl – a SHA-3 candidate, 2008.
6. H. Gilbert and T. Peyrin, Super-Sbox Cryptanalysis: Improved Attacks for AES-like permutations, FSE 2010, 2010.
7. L. R. Knudsen, Truncated and Higher Order Differentials, FSE 1994, LNCS 1008, pp. 196-211, 1994.
8. D. E. Knuth, The Art of Computer Programming - Seminumerical Algorithms, vol. 2, 3rd ed., Addison-Wesley, 1997.
9. M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen, and M. Schläffer, Rebound Distinguishers: Results on the Full Whirlpool Compression Function, ASIACRYPT 2009, LNCS 5912, pp. 126-143, 2009.
10. M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen, and M. Schläffer, The Rebound Attack and Subspace Distinguishers: Application to Whirlpool, Cryptology ePrint Archive: Report 2010/198.
11. F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen, The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl, FSE 2009, LNCS 5665, pp. 260-276, 2009.
12. F. Mendel, T. Peyrin, C. Rechberger, and M. Schläffer, Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher, SAC 2009, LNCS 5867, pp. 16-35, 2009.
13. F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen, Rebound Attacks on the Reduced Grøstl Hash Function, CT-RSA 2010, LNCS 5985, pp. 350-365, 2010.
14. 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.
15. T. Peyrin, Improved Differential Attacks for ECHO and Grøstl, Cryptology ePrint Archive: Report 2010/223.
16. T. Peyrin, Cryptanalysis of Grindahl, ASIACRYPT 2007, LNCS 4833, pp. 551-567, 2008.
17. X. Wang, Y. L. .Yin, and H. Yu, "Finding Collisions in the Full SHA-1," CRYPTO 2005, LNCS 3621, pages 17-36, Springer, 2005.

A Additional Paths Used in the Start-from-the-Middle Rebound Technique

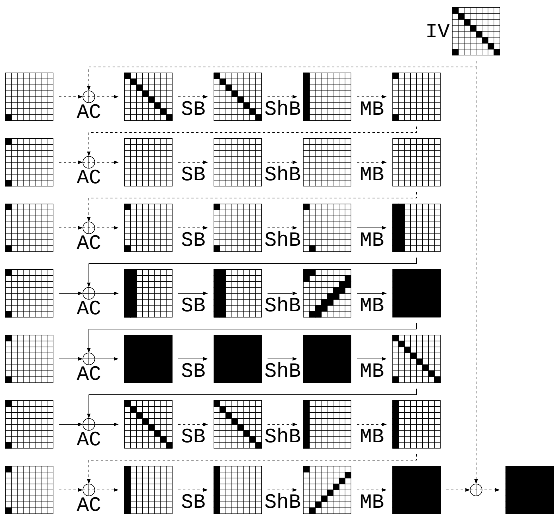

Fig. 10. Differential path between P and Q of 7-round Grøstl-224 and -256

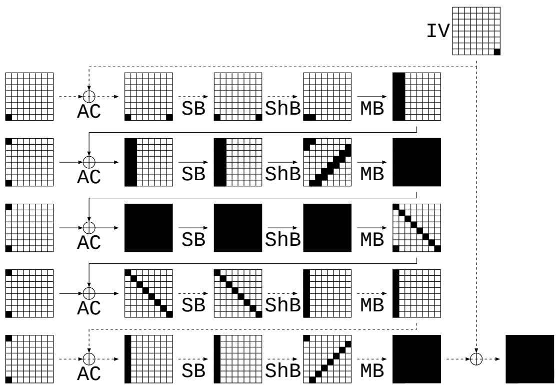

Fig. 11. Differential path between P and Q of 5-round Grøstl-224