The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl

Florian Mendel[1] and Christian Rechberger[1] and Martin Schläffer[1] and Søren S. Thomsen[2]

[1] Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
[2] Department of Mathematics, Technical University of Denmark
Matematiktorvet 303S, DK-2800 Kgs. Lyngby, Denmark
martin.schlaeffer@iaik.tugraz.at

Abstract. In this work, we propose the rebound attack, a new tool for the cryptanalysis of hash functions. The idea of the rebound attack is to use the available degrees of freedom in a collision attack to efficiently bypass the low probability parts of a differential trail. The rebound attack consists of an inbound phase with a match-in-the-middle part to exploit the available degrees of freedom, and a subsequent probabilistic outbound phase. Especially on AES based hash functions, the rebound attack leads to new attacks for a surprisingly high number of rounds.
We use the rebound attack to construct collisions for 4.5 rounds of the 512-bit hash function Whirlpool with a complexity of 2^{120} compression function evaluations and negligible memory requirements. The attack can be extended to a near-collision on 7.5 rounds of the compression function of Whirlpool and 8.5 rounds of the similar hash function Maelstrom. Additionally, we apply the rebound attack to the SHA-3 submission Grøstl, which leads to an attack on 6 rounds of the Grøstl-256 compression function with a complexity of 2^{120} and memory requirements of about 2^{64}.

Keywords: Whirlpool, Grøstl, Maelstrom, hash function, collision attack, near-collision

1 Introduction

In the last few years the cryptanalysis of hash functions has become an important topic within the cryptographic community. Especially the attacks and tools for the MD4 family of hash functions (e.g. MD5, SHA-1) have reduced the security provided by these commonly used hash functions [2, 3, 4, 24, 26, 27]. Most of the existing cryptanalytic work has been published for this particular line of hash function design. In the NIST SHA-3 competition [19], whose aim is to find an alternative hash function to SHA-2, many new hash function designs have been proposed. This is the most recent and most prominent case showing that it is very important to have tools available to analyze other design variants as well. Our work contributes to this toolbox.

1.1 Preview of Results

Our main result is the introduction of a technique for hash function cryptanalysis, which we call the *rebound attack*. We apply it to both block cipher based and permutation based constructions. In the rebound attack, we consider the internal cipher of a hash or compression function as three sub-ciphers. Let E be a block cipher, then $E = E_{fw} \circ E_{in} \circ E_{bw}$. Alternatively, for a permutation based construction, we decompose a permutation P into three sub-permutations $P = P_{fw} \circ P_{in} \circ P_{bw}$.

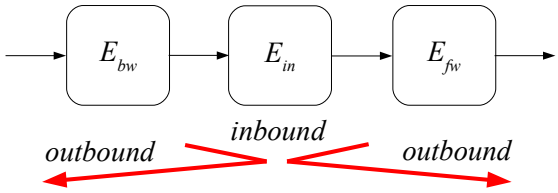

Fig. 1. A schematic view of the rebound attack. The attack consists of an inbound and two outbound phases.

The rebound attack can be described by two phases (see Fig. 1):

- **Inbound phase**: Is a meet-in-the-middle phase in E_{in} (or P_{in}), which is aided by the degrees of freedom that are available to a hash function cryptanalyst. We term the combination of meet-in-the-middle technique and exploitation of degrees of freedom leading to very efficient matches **match-in-the-middle approach**.
- **Outbound phase**: In this second phase, we use truncated differentials in both forward- and backward direction through E_{fw} and E_{bw} (or P_{fw} and P_{bw}) to obtain desired collisions or near-collisions. If the truncated differentials have a low probability in E_{fw} and E_{bw}, we can repeat the inbound phase to obtained more starting points for the outbound phase.

We apply the rebound attack on several concrete hash functions where the application on Whirlpool is probably the most relevant. Whirlpool is the only hash function standardized by ISO/IEC 10118-3:2003 (since 2000) that does not follow the MD4 design strategy. Furthermore, Whirlpool has been evaluated and approved by NESSIE [20]. Whirlpool is commonly considered to be a conservative block-cipher based design with an extremely conservative key schedule. The employed wide-trail design strategy [5] makes the application of differential and linear attacks seemingly impossible. No cryptanalytic results on the hash function Whirlpool have been published since its proposal 8 years ago.

Offsprings of Whirlpool are Maelstrom and to some extent several SHA-3 candidates, including `Grøstl`. The results of the attack on these hash functions are summarized in Table 1. For the types of attacks, we adopt the notation of [15].

Table 1. Summary of results of the attacks on reduced hash functions Whirlpool, Grøstl-256 and Maelstrom. The full versions have 10 rounds each. All attacks, except the attacks on Grøstl-256, have negligible memory requirements.

hash function	rounds	computational complexity	memory requirements	type	section
Whirlpool	4.5	2^{120}	2^{16}	collision	3
	5.5	2^{120}	2^{16}	semi-free-start collision	3
	7.5	2^{128}	2^{16}	semi-free-start near-collision	3
Grøstl-256	6	2^{120}	2^{64}	semi-free-start collision	4
Maelstrom	6.5	2^{120}	2^{16}	free-start collision	A
	8.5	2^{128}	2^{16}	free-start near-collision	A

1.2 Related Work

The rebound attack can be seen to have ancestors from various lines of research, often related to block ciphers:

- Firstly, differential cryptanalysis of block cipher based hash functions. Rijmen and Preneel [23] describe collision attacks on 15 out of 16 rounds on hash functions using DES. For the case of Whirlpool, there is an observation on the internal block cipher W by Knudsen [13]. Khovratovich *et al.* [11] studied collision search for AES-based hash functions.
- Secondly, inside-out techniques. As an application of second order differential attacks, inside-out techniques in block-cipher cryptanalysis were pioneered by Wagner in the Boomerang attack [25].
- Thirdly, truncated differentials. In the applications of the rebound technique, we used truncated differentials in the outbound parts. Knudsen [12] proposed truncated differentials as a tool in block cipher cryptanalysis, which recently have been applied to the hash function proposal Grindahl [14] by Peyrin [21].

1.3 Outline of the Paper

In the following section, we start with a description of the attacked hash functions. For the sake of presentation and concreteness, we immediately apply the rebound attack to the hash function Whirlpool in Sect. 3. In Sect. 4, we apply the rebound attack on Grøstl. The application of the attack to Maelstrom is postponed to App. A. We conclude in Sect. 5.

2 Description of the Hash Functions

In this section we give a short description of the hash functions to be analyzed in the remainder of this paper. We describe the hash function Whirlpool first, and continue with the description of the hash function Grøstl.

2.1 The Whirlpool Hash Function

Whirlpool is a cryptographic hash function designed by Barreto and Rijmen in 2000 [1]. It is an iterative hash function that processes 512-bit input message blocks with compression functions and produces a 512-bit hash value. The Whirlpool compression function basically consists of two parts: the key schedule and the state update transformation. The underlying block cipher W operates in the Miyaguchi-Preneel mode [17] as shown in Fig. 2. A detailed description of the hash function is given in [1].

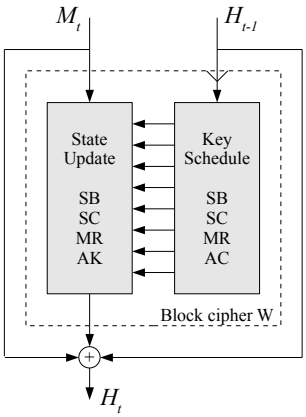

Fig. 2. A schematic view of the Whirlpool compression function. The block cipher W is used in Miyaguchi-Preneel mode.

The 512-bit block cipher W uses a 512-bit key and is similar to the Advanced Encryption Standard (AES) [18]. Both the state update transformation and the key schedule of W update an 8×8 state S of 64 bytes in 10 rounds each. The round transformations are very similar to the AES round transformations and are briefly described here:

- the non-linear layer SubBytes (SB) applies an S-Box to each byte of the state independently
- the cyclical permutation ShiftColumns (SC) rotates the bytes of column j downwards by j positions
- the linear diffusion layer MixRows (MR) multiplies the state by a constant matrix
- the key addition AddRoundKey (AK) adds the round key and/or the round constants c^r (AC) of the key schedule

In each round, the state is updated by round transformation r_i as follows:

$$r_i \equiv AK \circ MR \circ SC \circ SB.$$

In the remainder of this paper, we will use the outline of Fig. 3 for one round. We denote the resulting state of round transformation r_i by S_i and the intermediate states after SubBytes by S'_i, after ShiftColums by S''_i and after MixRows by S'''_i. The initial state prior to the first round is denoted by S_0.

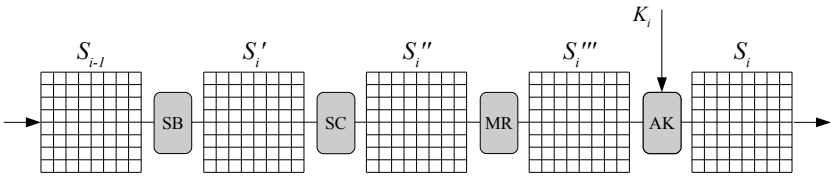

Fig. 3. One round r_i of the Whirlpool compression function with 8×8 states S_{i-1}, S'_i, S''_i, S'''_i, S_i and round key input K_i.

After the last round of the state update transformation, the initial value or previous chaining value $H_{t-1} = S_0$, the message block M_t, and the output value of the last round S_{10} are XORed, resulting in the final output of the Whirlpool compression function, $H_t = H_{t-1} \oplus M_t \oplus S_{10}$.

2.2 The Grøstl Hash Function

Grøstl was proposed by Gauravaram *et al.* as a candidate for the SHA-3 competition [9], initiated by the National Institute of Standards and Technology (NIST). Grøstl is an iterated hash function with a compression function built from two distinct permutations (see Fig. 4). Grøstl is a wide-pipe design with proofs for the collision and preimage resistance of the compression function [8].

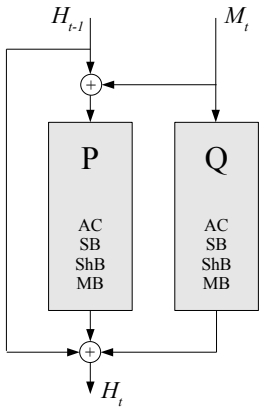

Fig. 4. The compression function of Grøstl. P and Q are $2n$-bit permutations for an n-bit hash value.

The two permutations P and Q are constructed using the wide trail design strategy and borrow components from the AES. The design of the two permutations is very similar to the block cipher W used in Whirlpool instantiated with a fixed key input. Both permutations update an 8×8 state of 64 bytes in 10 rounds each. The round transformations are very similar to the AES round transformations and are briefly described here:

- AddRoundConstant (AC) adds different one-byte round constants to the 8×8 states of P and Q
- the non-linear layer SubBytes (SB) applies the AES S-Box to each byte of the state independently
- the cyclical permutation ShiftBytes (ShB) rotates the bytes of row j left by j positions
- the linear diffusion layer MixBytes (MB) multiplies the state by a constant matrix

In each round, the state is updated by round transformation r_i as follows:

$$r_i \equiv MB \circ ShB \circ SB \circ AC$$

3 Rebound Attack on Whirlpool

In this section, we present details of the rebound attacks applied to the hash function Whirlpool. First, we will give an overview of the attack strategy which is the basis for the attacks on 4.5, 5.5 and 7.5 rounds. The main idea of the attacks is to use a 4-round differential trail [6], which has the following sequence of active S-boxes: $1 \rightarrow 8 \rightarrow 64 \rightarrow 8 \rightarrow 1$. Note that the differential probability in each round is proportional to the number of active S-boxes. Using the Rebound Attack we can cover the most expensive middle part using an efficient match-in-the-middle approach (inbound phase). In the outbound phase, the trail is extended and the two ends of the trail are linked using the feed-forward of the hash function.

3.1 Attack Overview

The core of the attack is a 4 round trail of the form $1 \rightarrow 8 \rightarrow 64 \rightarrow 8 \rightarrow 1$. This trail has the minimum number of active S-boxes and has the best differential probability according to the wide trail design strategy. In the rebound attack, we first split the block cipher W into three sub-ciphers $W = E_{fw} \circ E_{in} \circ E_{bw}$, such that most expensive part of the differential trail is covered by the efficient inbound phase E_{in}. Then, the outbound phase (E_{fw}, E_{bw}) has a relatively low probability and can be fulfilled in a probabilistic way:

$$\begin{aligned} E_{bw} &= SC \circ SB \circ AK \circ MR \circ SC \circ SB \\ E_{in} &= MR \circ SC \circ SB \circ AK \circ MR \\ E_{fw} &= AK \circ MR \circ SC \circ SB \circ AK \end{aligned}$$

The two phases of the rebound attack consists of basically four steps:

– **Inbound phase**
  Step 1: start with 8-byte truncated differences at the MixRows layer of round r_2 and r_3, and propagate forward and backward to the S-box layer of round r_3.
  Step 2: connect the input and output of the S-boxes of round r_3 to form the three middle states $8 \rightarrow 64 \rightarrow 8$ of the trail.
– **Outbound phase**
  Step 3: extend the trail both forward and backward to give the trail $1 \rightarrow 8 \rightarrow 64 \rightarrow 8 \rightarrow 1$ through MixRows in a probabilistic way.
  Step 4: link the beginning and the end of the trail using the feed-forward of the hash function.

If the differences in the first and last step are identical, they cancel each other through the feed-forward. The result is a collision of the round-reduced compression function of Whirlpool. See Fig. 5 for an overview of the attack.

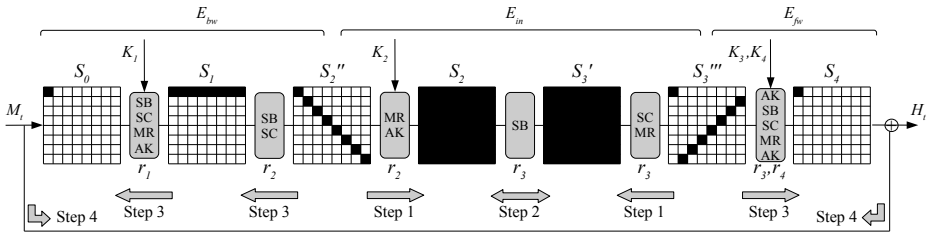

Fig. 5. A schematic view of the attack on 4 rounds of Whirlpool with round key inputs and feed-forward. Black state bytes are active.

3.2 Collision Attack for 4.5 Rounds

The collision attack on 4.5 rounds of Whirlpool is the starting point for all subsequent attacks. If the differences in the message words are the same as in the output of the state update transformation, the differences cancel each other through the feed-forward. In other words, we will construct a fixed-point (in terms of differences) for the block cipher in the state update. The outline of the attack is shown in Fig. 5 and the sequence of truncated differences has the form:

$$1 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8 \xrightarrow{r_4} 1 \xrightarrow{r_{4.5}} 1$$

In the following, we analyze the 4 steps of the attack in detail.

Precomputation. In the match-in-the-middle part (Step 2) we need to find a differential match for the SubBytes layer. In a precomputation step, we compute a 256×256 lookup table for each S-box differential (input/output XOR difference table). Note that only about 1/2 of all S-box differentials exist. For each

possible S-box differential, there are at least two (different) values such that the differential holds. A detailed description of the distribution of S-box differentials is given in App. B.

Step 1. We start the attack by choosing a random difference with 8 active bytes of state S_2'' prior to the MixRows layer of round r_2. Note that all active bytes have to be in the diagonal of state S_2'' (see Fig. 5). Then, the differences propagate forward to a full active state at the input of the next SubBytes layer (state S_2) with a probability of 1. Next, we start with another difference and 8 active bytes in state S_3''' after the MixRows transformation of round r_3 and propagate backwards. Again, the diagonal shape ensures that we get a full active state at the output of SubBytes of round r_3.

Step 2. In Step 2, the match-in-the-middle step, we look for a matching input/output difference of the SubBytes layer of round r_3 using the precomputed S-box differential table. Since we can find a match with a probability of $1/2$ for each byte, we can find a differential for the whole active SubBytes layer with a probability of about 2^{-64}. Hence, after repeating Step 1 of the attack about 2^{64} times, we expect to find a SubBytes differential for the whole state. Since we get at least two state values for each S-box match, we get about 2^{64} starting points for the outbound phase. Note that these 2^{64} starting points can be constructed with a total complexity of about 2^{64}. In other words, the average computational cost of each match-in-the-middle step is essentially the respective computation of the round transformations.

Step 3. In the outbound phase, we further extend the differential path backward and forward. By propagating the matching differences and state values through the next SubBytes layer, we get a truncated differential in 8 active bytes for each direction. Next, the truncated differentials need to follow a specific active byte pattern. In the case of the 4 round Whirlpool attack, the truncated differentials need to propagate from 8 to one active byte through the MixRows transformation, both in the backward and forward direction.

The propagation of truncated differentials through the MixRows transformation is modelled in a probabilistic way. The transition from 8 active bytes to one active byte through the MixRows transformation has a probability of about 2^{-56} (see App. C). Note that we require a specific position of the single active byte to find a match in the feed-forward (Step 4). Since we need to fulfill one $8 \rightarrow 1$ transitions in the backward and forward direction, the probability of the outbound phase is $2^{-2 \cdot 56} = 2^{-112}$. In other words, we have to repeat the inbound phase about 2^{112} times to generate 2^{112} starting points for the outbound phase of the attack.

Step 4. To construct a collision at the output of this 4 round compression function, the exact value of the input and output difference has to match. Since

only one byte is active, this can be fulfilled with a probability of 2^{-8}. Hence, the complexity to find a collision for 4 rounds of Whirlpool is $2^{112+8} = 2^{120}$. Note that we can add half of a round (SB,SC) at the end for free, since we are only interested in the number of active bytes. Remember that we can construct up to 2^{128} starting points in the inbound phase of the attack, hence we have enough degrees of freedom for the attack. Note that the values of the key schedule are not influenced. Hence, the attack works with the standard IV and we can construct collisions for 4.5 rounds of the hash function of Whirlpool.

3.3 Semi-Free-Start Collision Attack for 5.5 Rounds

We can extend the collision attack on 4.5 rounds to a semi-free-start collision attack on 5.5 rounds of Whirlpool. The idea is to add another full active state in the middle of the trail. We use the additional degrees of freedom of the key schedule to fulfill the difference propagation through *two* full active S-box transformations. Note that the outbound part of the attack stays the same and the new sequence of active S-boxes is:

$$1 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 64 \xrightarrow{r_4} 8 \xrightarrow{r_5} 1 \xrightarrow{r_{5.5}} 1$$

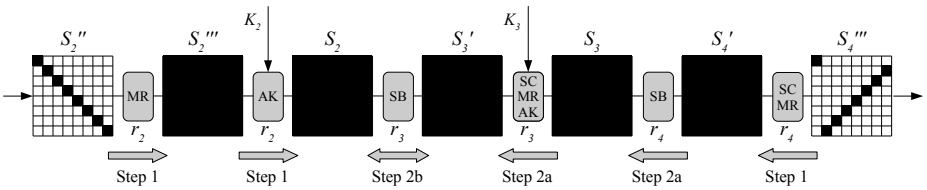

Fig. 6. In the attack on 5.5 rounds we first choose random values of the state S_4' to propagate backwards (Step 2a) and then, use the degrees of freedom from the key schedule to solve the difference propagation of the S-box in round r_3 (Step 2b).

Step 1. Figure 6 shows the inbound part of the attack in detail. Again, we can choose from up to 2^{64} initial differences with 8 active bytes at state S_2'' and S_4''' each, and linearly propagate forward to S_2 and backward to S_4' until we hit the first S-box layer. Then, we need to find a matching SubBytes differential of two consecutive S-box layers in the match-in-the-middle phase.

Step 2. To pass the S-box of round r_4 in the backward direction, we choose one of 2^{512} possible values for state S_4'. This also determines the input values and differences of the SubBytes layer (state S_3). Then, we propagate the difference further back to state S_3', which is the output of the S-box in round r_3. The 512

degrees of freedom of the key schedule input K_3 between the two S-boxes allow us to still assign arbitrary values to the state S'_3. Hence, the correct difference propagation of the S-box in round r_3 can be fulfilled by using these additional degrees of freedom to choose the state S'_3 as well. The complexity of the attack does not change and is determined by the 2^{120} trials of the outbound phase.

The outbound phase (Step 3 and Step 4) of the 5.5 round attack is equivalent to the 4.5 round case. However, we cannot choose the round keys, and hence the chaining values, anymore since they are determined by the difference propagation of the S-box of round r_3. Therefore, this 5.5 round attack is only a semi-free-start collision attack on the hash function of Whirlpool.

3.4 Semi-Free-Start Near-Collision Attack for 7.5 Rounds

The collision attack on 5.5 rounds can be further extended by adding one round at the beginning and one round and at the end of the trail (see Fig. 7). The result is a semi-free-start near-collision attack on 7.5 rounds of the hash function Whirlpool with the following number of active S-boxes:

$$8 \xrightarrow{r_1} 1 \xrightarrow{r_2} 8 \xrightarrow{r_3} 64 \xrightarrow{r_4} 64 \xrightarrow{r_5} 8 \xrightarrow{r_6} 1 \xrightarrow{r_7} 8 \xrightarrow{r_{7.5}} 8$$

Since the inbound phase (Step 1 and Step 2) is identical to the attack on 5.5 rounds, we only discuss the outbound phase (Step 3 and Step 4) here.

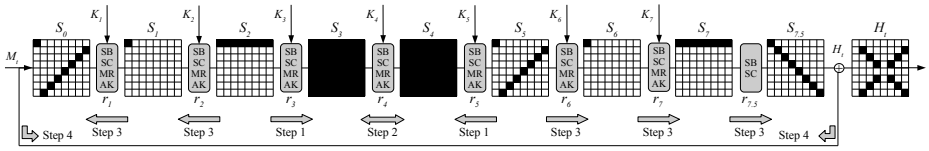

Fig. 7. In the attack on 7.5 rounds we extend the trail by one more round at the beginning and 1.5 rounds at the end to get a semi-free-start near-collision of Whirlpool.

Step 3. The 1-byte difference at the beginning and end of the 4 round trail will always result in 8 active bytes after one MixRows transformation. Hence, we can go backward one round and forward 1.5 rounds with no additional costs. We add a half round at the end to get a similar pattern of 8 active S-boxes due to the ShiftColumns transformation. Note that we cannot get an exact match of active S-boxes and get therefore only a semi-free-start near-collision.

Step 4. Using the feed-forward, the position of two active S-boxes match and cancel each other with a probability of 2^{-16}. Hence, the total complexity of our semi-free-start near-collision is about $2^{112+16} = 2^{128}$. Note that the generic (birthday) complexity of a near-collision on 52 bytes is $2^{\frac{52 \cdot 8}{2}} = 2^{208}$.

4 Rebound Attack on Grøstl

In this section, we extend the attack on Whirlpool to the SHA-3 proposal Grøstl. Although the hash function is built from similar components as Whirlpool, the attack does not apply equally well. The available degrees of freedom of the second permutation cannot be used in the attack on the first permutation as in Whirlpool. Note that we can still apply the attack on 4.5 rounds of Whirlpool to the compression function of Grøstl-256 and get the same complexity of about 2^{120}.

4.1 Semi-Free-Start Collision for 5 Rounds

We can improve the Rebound Attack on Grøstl-256 by using differences in the second permutation as well. In the attack on 5 rounds, we use the following differential trail for both permutations:

$$8 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8 \xrightarrow{r_4} 8 \xrightarrow{r_5} 64$$

By using an equivalent differential trail in the second permutation one can find a collision for the compression function of Grøstl-256 reduced to 5 rounds with a complexity of 2^{64}, see Fig. 8.

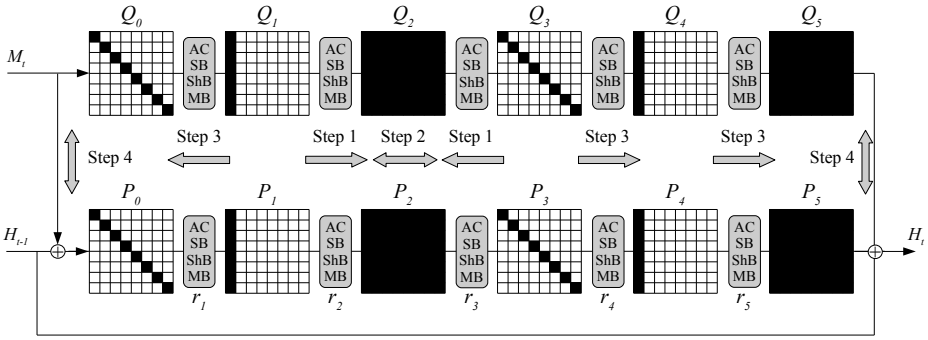

Fig. 8. Attack on Grøstl-256 reduced to 5 rounds using two equivalent trails in both permutations P and Q.

For each permutation, we can find 2^{64} inputs following this differential with a complexity of about 2^{64} and negligible memory requirements, see Sect. 3. Hence, the differential trail holds with probability 1 on average in both P and Q. In order to get a semi-free-start collision of Grøstl-256 reduced to 5 rounds, we require that the differences at the output of round 5 are equal. Since the MixBytes transformation is linear it is sufficient that the differences before MixBytes in round 5 are equal. Furthermore, to prevent that the feed-forward destroys the collision again, we do not allow any differences in H. Hence, all differences are

due to differences in the message M and we require these differences at the input of round 1 to be equal as well.

For the attack to work, differences in 16 bytes need to be equal. A straightforward implementing of the attack would result in a complexity of about 2^{128}. However, the complexity can be significantly reduced by applying a meet-in-the-middle attack. In detail, by generating 2^{64} differential trails for P and 2^{64} differential trails for Q we expect to find a matching input and output. This results in a semi-free-start collision for Grøstl-256 reduced to 5 rounds. The attack has a total complexity of about 2^{64} evaluations of P and Q and memory requirement of 2^{64}. Note that the memory requirements of the attack can be significantly reduced by memory less variants of the meet-in-the-middle attack [22].

4.2 Semi-Free-Start Collision for 6 Rounds

The attack can be extended to 6 rounds using an extended differential trail for P and Q, see Fig. 9. For this attack, we use a trail with the following sequence of active bytes:

$$8 \xrightarrow{r_1} 1 \xrightarrow{r_2} 8 \xrightarrow{r_3} 64 \xrightarrow{r_4} 8 \xrightarrow{r_5} 8 \xrightarrow{r_6} 64$$

Note that this trail holds with a probability of 2^{-56} on average. Hence, we can find a collision for the compression function of Grøstl-256 reduced to 6 rounds with a complexity of about $2^{56+64} = 2^{120}$, and memory requirements of 2^{64} to match the beginning and end of each trail. In contrast to the attack on 5 rounds, we do not see how the connection of the two permutations can be implemented in a memory-less way.

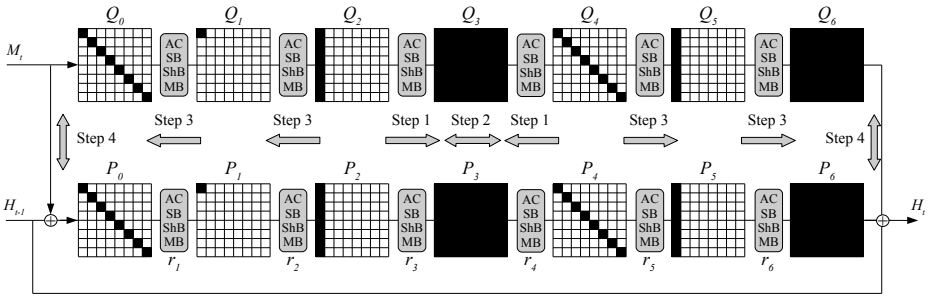

Fig. 9. Attack on Grøstl-256 reduced to 6 rounds using two equivalent trails in both permutations P and Q.

Note that we could add a half round (ShB,MB) in the beginning of Grøstl-256, similar to the end of the trail. However, we only consider variants by reducing rounds at the end of the compression function. Trying to attack more rounds of the Grøstl-256 compression function quickly does not leave enough degrees of freedom to succeed, or results in a computational complexity above 2^{128}, which is above the security claims of the designers.

5 Conclusion and Open problems

In this paper, we propose a new tool for the toolbox of hash function cryptanalysts: The rebound attack. We have successfully attacked 7.5 rounds of the Whirlpool compression function, 6 rounds of the `Grøstl`-256 compression function, and 8.5 rounds of the Maelstrom compression function (App. A).

The idea in these attacks is to use the available degrees of freedom in a collision attack to efficiently bypass the devastating effects of the wide-trail design strategy on differential-style attacks for a feasible number of rounds. More degrees of freedom (like the increased key-size in the Maelstrom block cipher) makes equation solving (and hence the match-in-the-middle step) easier and allows to cover even more rounds.

Most AES-based SHA-3 candidates are natural candidates for applications of the rebound attack. To this end, we can refer to preliminary results which break Twister-512 [16][3].

The idea seems applicable to a wider range of hash function constructions. For the outbound part of the rebound attack we used truncated differentials in all our examples. However, the rebound technique does not constrain the property used in the outbound part. It would be interesting to see if other non-random properties (e.g., correlations or algebraic relations) could also be used with the rebound attack.

Acknowledgments. We would like to thank Henri Gilbert, Mario Lamberger, Tomislav Nad, Vincent Rijmen and the anonymous referees for useful comments and discussions. The work in this paper has been supported in part by the Secure Information Technology Center-Austria (A-SIT), by the European Commission under contract ICT-2007-216646 (ECRYPT II) and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

References

1. Paulo S. L. M. Barreto and Vincent Rijmen. The Whirlpool Hashing Function. Submitted to NESSIE, September 2000. Revised May 2003. Available online at `http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html` (2008/12/11).
2. Christophe De Cannière, Florian Mendel, and Christian Rechberger. Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *LNCS*, pages 56–73. Springer, 2007.
3. Christophe De Cannière and Christian Rechberger. Finding SHA-1 Characteristics: General Results and Applications. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.

[3] A variant of the match-in-the-middle part of the rebound attack led to practical attacks on the compression function, which in turn led to theoretical attacks on the Twister hash function.

4. Christophe De Cannière and Christian Rechberger. Preimages for Reduced SHA-0 and SHA-1. In David Wagner, editor, *CRYPTO*, volume 5157 of *LNCS*, pages 179–202. Springer, 2008.
5. Joan Daemen and Vincent Rijmen. The Wide Trail Design Strategy. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *LNCS*, pages 222–238. Springer, 2001.
6. Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Information Security and Cryptography. Springer, 2002. ISBN 3-540-42580-2.
7. Decio Gazzoni Filho, Paulo S.L.M. Barreto, and Vincent Rijmen. The Maelstrom-0 hash function. In *SBSeg 2006*, 2006.
8. Pierre-Alain Fouque, Jacques Stern, and Sébastien Zimmer. Cryptanalysis of Tweaked Versions of SMASH and Reparation. In *SAC*, LNCS. Springer, 2008. To appear.
9. Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl – a SHA-3 candidate. Available online at `http://www.groestl.info`, 2008.
10. L. Keliher and J. Sui. Exact Maximum Expected Differential and Linear Probability for Two-Round Advanced Encryption Standard. *Information Security, IET*, 1(2):53–57, June 2007.
11. Dmitry Khovratovich, Alex Biryukov, and Ivica Nikolic. Speeding up collision search for byte-oriented hash functions. In *CT-RSA*, 2009. To appear.
12. Lars R. Knudsen. Truncated and Higher Order Differentials. In Bart Preneel, editor, *FSE*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.
13. Lars R. Knudsen. Non-random properties of reduced-round Whirlpool. NESSIE public report, NES/DOC/UIB/WP5/017/1, 2002.
14. Lars R. Knudsen, Christian Rechberger, and Søren S. Thomsen. The Grindahl Hash Functions. In Alex Biryukov, editor, *FSE*, volume 4593 of *LNCS*, pages 39–57. Springer, 2007.
15. Xuejia Lai and James L. Massey. Hash Function Based on Block Ciphers. In *EUROCRYPT*, pages 55–70, 1992.
16. Florian Mendel, Christian Rechberger, and Martin Schläffer. Cryptanalysis of Twister. In Michel Abdalla and David Pointcheval, editors, *ACNS*, volume 5536 of *LNCS*. Springer, 2009. To appear.
17. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. Available online at `http://www.cacr.math.uwaterloo.ca/hac/`.
18. National Institute of Standards and Technology. FIPS PUB 197, Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, U.S. Department of Commerce, November 2001.
19. National Institute of Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. *Federal Register*, 27(212):62212–62220, November 2007. Available: `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf` (2008/10/17).
20. NESSIE. New European Schemes for Signatures, Integrity, and Encryption. IST-1999-12324. Available online at `http://cryptonessie.org/`.
21. Thomas Peyrin. Cryptanalysis of Grindahl. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *LNCS*, pages 551–567. Springer, 2007.
22. Jean-Jacques Quisquater and Jean-Paul Delescaille. How easy is collision search? Application to DES (extended summary). In Jean-Jacques Quisquater and Joos

Vandewalle, editors, *EUROCRYPT*, volume 434 of *LNCS*, pages 429–434. Springer, 1989.
23. Vincent Rijmen and Bart Preneel. Improved Characteristics for Differential Cryptanalysis of Hash Functions Based on Block Ciphers. In Bart Preneel, editor, *FSE*, volume 1008 of *LNCS*, pages 242–248. Springer, 1994.
24. Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *LNCS*, pages 1–22. Springer, 2007.
25. David Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.
26. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
27. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

A Rebound Attack on Maelstrom

In this section, we apply the attack of Whirlpool to Maelstrom.

A.1 Description of the Hash Function

Maelstrom [7] is a hash function very similar to Whirlpool. It has a simpler key schedule, works on 1024-bit message blocks and uses the Davies-Meyer mode instead of Miyaguchi-Preneel. The internal block cipher of Maelstrom works on 512-bit blocks with a 1024-bit key schedule. The additional 512 degrees of freedom in the key schedule can be used to attack one more round (up to 8.5 rounds) of the compression function of Maelstrom.

A.2 Attack on 8.5 Rounds

Since Maelstrom uses the Davies-Meyer mode, we can only get a free-start collision for the hash function. However, the additional degrees of freedom of the key schedule allow us to add another round in the inbound part. The sequence of active S-boxes for the 8.5 round attack on Maelstrom is then:

$$8 \xrightarrow{r_1} 1 \xrightarrow{r_2} 8 \xrightarrow{r_3} 64 \xrightarrow{r_4} 64 \xrightarrow{r_5} 64 \xrightarrow{r_6} 8 \xrightarrow{r_7} 1 \xrightarrow{r_8} 8 \xrightarrow{r_{8.5}} 8 \tag{1}$$

The extension is essentially the same as for the 7.5 round attack on Whirlpool. We add another state with 64 active bytes in the middle of the trail. This means, that we now have to fulfill the difference propagation of three S-box layers with 64 active bytes each. Same as in Sect. 3.3, we can fulfill one S-box propagation using the 512 degrees of freedom of the state itself. Since the second S-box difference propagation uses only 512 degrees of freedom from the key schedule, there are another 512 degrees of freedom left to fulfill the difference propagation of the third S-box. The complexity of the attack does not change and is 2^{128} for

the 512-bit hash function Maelstrom. Furthermore, the semi-free-start collision attack on 5.5 rounds of Whirlpool can be extended to a 6.5 rounds free-start collision attack of Maelstrom with the same complexity of 2^{120}.

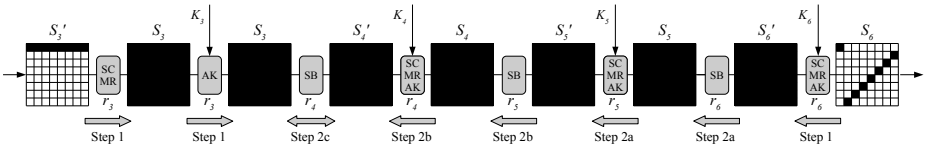

Fig. 10. In the attack on Maelstrom we use the additional degrees of freedom of the key schedule to pass three S-box layers (Step 2a,2b,2c).

B Probability and Conditions of S-box Differentials

In this section we analyze differentials of the Whirlpool and AES S-boxes in detail. For a fixed differential $(\Delta a, \Delta b)$ with $\Delta a = x \oplus y$ and $\Delta b = S(x) \oplus S(y)$, we get the probability $P(\Delta b = S(\Delta a)) \sim 1/2$. This can be verified by enumerating through all 256×256 input/output pairs (x, y) and $(S(x), S(y))$. Table 2 gives a distribution of possible S-box differentials for the Whirlpool and AES S-boxes [10]. Note that for each *possible* S-box differential, we get at least the two symmetric values (x, y) and (y, x). In the case of Whirlpool, we get for a small fraction of differentials even 8 possible pairs. This corresponds to the maximum probability distribution of the Whirlpool S-box, which is $8 \cdot 2^{-8} = 2^{-5}$.

Table 2. The number of differentials and possible pairs (x, y) for the Whirlpool and AES S-boxes. The first row shows the number of impossible differentials and the last row corresponds to the zero differential.

# (x, y)	Whirlpool	AES
0	39655	33150
2	20018	32130
4	5043	255
6	740	-
8	79	-
256	1	1

C Propagation of Truncated Differentials in MixRows and MixBytes

Since the MixRows operation is a linear transformation, standard differences propagate through MixRows in a deterministic way. The propagation only de-

pends on the values of the differences and is independent of the actual value of the state. In case of truncated differences only the position, but not the value of the difference is determined. Therefore, the propagation of truncated differences through MixRows can only be modelled in a probabilistic way. Note that the MixBytes operation of `Grøstl` has the same properties as MixRows.

The MDS property of the MixRows transformation ensures that the sum of the number of active input and output bytes is at least 9. Hence, a non-zero truncated difference with one active byte will propagate to a truncated difference with 8 active bytes with a probability of 1. On the other hand, a truncated difference with 8 active bytes can result in a truncated difference with one to 8 active bytes after MixRows. However, the probability of a $8 \rightarrow 1$ transition with predefined positions is only $2^{-7 \cdot 8} = 2^{-56}$ since we require 7 out of 8 truncated differences to be zero. Table 3 is similar to the table of [21] and shows the probabilities for all 81 cases with a fixed position of truncated differences. Note that the probability of any $x \rightarrow 8$ transition $(1 - \sum_{i=1}^{7} P(x \rightarrow i) \sim 2^{-0.0017})$ is approximated by 1 in this paper. Note that the probability only depends on the direction of the *propagation* of truncated differences.

Table 3. Approximate probabilities for the propagation of truncated differences through MixRows with predefined positions. D_i denotes the number of active bytes at the input and D_o the number of active bytes at the output of MixRows. Probabilities are base 2 logarithms.

$D_o \setminus D_i$	0	1	2	3	4	5	6	7	8
0	0	×	×	×	×	×	×	×	×
1	×	×	×	×	×	×	×	×	−56
2	×	×	×	×	×	×	×	−48	−48
3	×	×	×	×	×	×	−40	−40	−40
4	×	×	×	×	×	−32	−32	−32	−32
5	×	×	×	×	−24	−24	−24	−24	−24
6	×	×	×	−16	−16	−16	−16	−16	−16
7	×	×	−8	−8	−8	−8	−8	−8	−8
8	×	0	-0.0017	-0.0017	-0.0017	-0.0017	-0.0017	-0.0017	-0.0017