Cryptanalysis of hash functions with structures

Dmitry Khovratovich

University of Luxembourg

{dmitry.khovratovich@uni.lu}

Abstract. Hash function cryptanalysis has acquired many methods, tools and tricks from other areas, mostly block ciphers. In this paper another trick from block cipher cryptanalysis, the structures, is used for speeding up the collision search. We investigate the memory and the time complexities of this approach under different assumptions on the round functions. The power of the new attack is illustrated with the cryptanalysis of the hash functions Grindahl and the analysis of the SHA-3 candidate Fugue (both functions as 256 and 512 bit versions). The collision attack on Grindahl-512 is the first collision attack on this function.

Keywords: cryptanalysis, hash functions, SHA-3, truncated differentials, Grindahl, Fugue, structures.

1 Introduction

Since 1990 the MD family of hash functions and its successor SHA family have been most widely used data integrity primitives. In contrast with few cryptanalytic results in 90s recent attacks on MD5 [19], SHA-0 [14], and SHA-1 [5] encouraged the cryptographic community to look for more reliable components and then motivated the recent SHA-3 competition [16]. The Merkle-Damgård approach [8, 15] to build hash functions from compression functions, has lost a part of credit due to such generic attacks as multicollisions [11] and second-preimage search with expandable messages [12].

In contrast to generic attacks like multicollisions, which are applicable to hash functions with Merkle-Damgard strengthening, attacks on lower level *compression functions* are highly dependent on a particular proposal and can rarely be extended to other functions. Common ideas are mostly related to the notion of differentials since the fact that two different messages produce the same hash value (a *collision*) can be expressed as a zero difference in the output.

The idea of differentials comes from block cipher cryptanalysis and pioneering papers by Biham and Shamir [3]. As high probability differential characteristics were exploited in attacks on block ciphers as high probability zero-ending differential *trails* are used to find collisions for compression functions.

Since block cipher cryptanalysis is a highly developed topic, many cryptanalysts try to use the most efficient methods and tools in attacks on hash functions.

However, due to stronger requirements on the results of an attack only few of them were applied. The use of truncated differentials [17] is an example.

In this paper we investigate another tool from block cipher cryptanalysis: *structures*. A structure is originally a set of plaintexts that pairwise have some property (e.g., zero difference in particular bytes). Since the number of pairs with desired properties in a structure is much larger than the structure size, such constructions are widely used in order to save memory and time in attacks on block ciphers [4, 6].

Intuitively, structures might have been used in attacks on the hash functions built on block ciphers [18]. However, the authors are aware of only one such attack: a recent attack on Snefru [2], though Snefru does not directly fit the constructions from [18].

We have found that structures are especially useful in attacks on stream-based hash functions, where parts of a message can independently be controlled. We analyze the hash functions Grindahl and Fugue. For Grindahl-256, we improve the best known attack by Peyrin [17], while for Grindahl-512 this paper presents the first known collision attack. The hash function Fugue [10] is a strengthened successor to Grindahl, so we did not manage to break its security claims. However, our attack is substantially faster than a trivial internal-collision attack.

This paper is organized as follows. First we briefly explain how the use of structures reduce the cost of collision search. Then we investigate how structures follow the differential trail and collapse to pairs in some step (Section 3) so that the standard differential approach can be applied afterwards. We also derive the memory complexity of the attack.

Then we attack Grindahl and Fugue with structures (Section 4). The number of computations required ro find a collision for Grindahl-256 is reduced compared to the attack by Peyrin [17]. We also present the first collision attack on Grindahl-512 and the first external analysis of Fugue . In the Appendix the time complexity of the attack is estimated under different assumptions.

2 Idea in brief

In many attacks on compression functions a cryptanalyst deals with a set of pairs that are to follow a particular differential trail. Here the *trail* is a sequence of differences in the internal state of the hash function. (see [1] for a more formal approach). At some steps an adversary may vary a message part to be injected thus increasing the number of pairs that follow the trail (the attack by De Canniére and Rechberger on SHA-1 [5] is an example). If there is not enough freedom to satisfy round conditions, the number of candidate pairs tends to decrease. We show that this effect can be postponed if a differential trail allows to incorporate pairs into structures.

In order to distinguish the approach when a cryptanalytic deals with pairs from our approach we call the former one the *standard differential attack*. It is

also known as the *trail backtracking* [1]. Our attack is later called *the structural approach*, or the *structural attack*.

Now assume that the trail deals with truncated differentials, and the possible differences form a linear space R of differences. Then if a pair of states (S_1, S_2) fits the trail, and a pair (S_2, S_3) fits the trail, then the pair (S_1, S_3) fits the trail too. Such a group of states is called a *structure*.

Suppose at some step a structure of size Q enters the round with probability P. Then every state S_i will have a desired difference with PQ other states thus composing a smaller structure. Therefore the initial structure splits into $1/P$ smaller structures. If the structure collapses into separate pairs then the differential attack is launched.

Suppose there is now freedom from the message injection, i.e., for a pair of messages (M, M') there are V possibilities for M and D more possibilities for M' (VD pairs at all). So if a round differential has probability P then of T pairs about $V \cdot D \cdot P \cdot T$ pairs survive. See Figure 1 for the outline of the situation.

When we work with structures, the value of the injected message can be chosen freely only for the first state of a new structure, or the *leader state*; the messages injected to the other states should have a desired difference with a first one. Consequently, the message freedom results in structures of size $\approx D \cdot P \cdot Q$. The number of states remains the same; however, it can be increased if we take other states as leading ones.

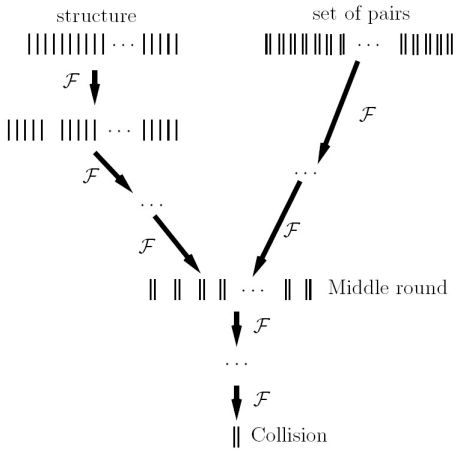

Fig. 1. Comparative view on the structural and the differential approaches. $\mathcal{F}$ is a round function. In the first case the number of states remains stable till structures collapse to pairs (*middle round*).

3 Analysis of structure fission

In order to benefit from the number of pairs in a structure an adversary should keep the size of the structure as big as possible. Let us estimate the size as a function of the round probability and the freedom given by message injections. Denote by $\sim$ the desired binary relation between two states, which can be also interpreted as the fact that the difference in the state satisfies the trail conditions. The particular relation is usually clear from the context. Then a *structure* is a set of internal states such that any two states of the set satisfy the binary relation $\sim$. In our attacks on Grindahl and Fugue (Section 4) the relation is of form "bytes (respectively, words) $i_1, i_2, \ldots, i_k$ are equal".

No freedom in the message injection. Suppose a structure of size $Q = 2^q$ states enters a round with probability 2^{-p}. First consider the case where there is no freedom in message injection due to the differential trail or the message schedule. After the application of the round function any state has a desired difference with 2^{q-p} other states, which form a structure of size 2^{q-p}. Therefore, the initial structure splits into about 2^p smaller structures of average size 2^{q-p}.

It is easy to prove that the partition of a structure into structures of equal size gives a lower bound on the overall number of pairs. If p is high enough then the structure collapses to separate pairs. Since there were 2^{2q-1} pairs in the structure, about 2^{2q-1-p} pairs come out of one round. Then the pairs are processed by the standard differential attack.

Value freedom in the message injection. Suppose now there is some freedom in message injections but the differential trail does not allow to introduce a difference, or the value of the difference is fixed. Then every state can be transformed to at most one element of a new structure, so the structures do not grow in this case. However, one may increase the number of structures and thus the number of considered states.

The latter approach increase the memory complexity so we do not use it except for the round when all structures collapse to pairs. Given $V = 2^v$ possibilities for an injected message we get $2^{2q-1+v-p}$ pairs after the round.

Difference freedom in the message injection. Assume that 2^d possible differences can be injected, and they form a linear space. Then we get larger structures because we have more freedom in steering a state into the structure.

Suppose that state S_i has already transformed to state S'_i by message v_i: $S_i[v_i] = S'_i$. Let us compute the probability that a randomly chosen state S_j can be transformed to some state with the desired difference with S'_i by some message m', which follows the trail too. The probability can be expressed as $\mathbb{P}(\exists m' : S_j[m'] \sim S'_i, m' \sim m \mid S_i \sim S_j)$.

Assuming that the events for particular messages are independent we obtain the following expression:

$$\mathbb{P}(\exists m' : S_j[m'] \sim S'_i, m' \sim m \mid S_i \sim S_j) = \\ = 1 - \prod_{m' \sim m} \mathbb{P}(S_j[m'] \nsim S'_i \mid S_i \sim S_j) = 1 - (1 - 2^{-p})^{2^d} \approx 2^{d-p}. \quad (1)$$

Consequently, one structure splits into structures of average size $Q' = 2^{q+d-p}$. Analogously, if $q + d - p < 0$ the structure collapse to pairs. Since $2^{2q-1+v+d}$ pairs can be composed about $2^{2q-1+v+d-p}$ come out of the round.

Size of the initial structure. By *degrees of freedom* we understand the base 2 logarithms of the number of admissible values. Suppose that at round i there are v_i degrees of freedom in the values of the injected message, d_i degrees of freedom in the differences in injected messages, and p_i (bit) conditions to be satisfied. In the standard differential attack we start with 2^c pairs and leave with one pair in the end. Therefore, we obtain the following equation:

$$c + \sum_{i=1}^{T} (v_i + d_i - p_i) = 0.$$

Here T stands for the number of rounds covered by the trail. We also denote by $c(t)$ the logarithm of the number of pairs after t-th round:

$$c(t) = c + \sum_{i=1}^{t} (v_i + d_i - p_i) = c + \underbrace{\sum_{i=1}^{t} v_i}_{v(t)} + \underbrace{\sum_{i=1}^{t} d_i}_{d(t)} - \underbrace{\sum_{i=1}^{t} p_i}_{p(t)}.$$

Suppose we start with a structure of size 2^q, which collapse to pairs after $l+1$ rounds, $l < T$. The structure splits to $2^{p(l)-d(l)}$ smaller structures after l rounds. Each structure is of size about $2^{q+d(l)-p(l)}$. Therefore, about $2^{2q+d(l)-p(l)-1}$ pairs come out of round l.

In order to continue the collision search and obtain one pair in the end the following equation should hold:

$$2q + d(l) - p(l) - 1 = c(l) \;\Leftrightarrow\; 2q = c + v(l) + 1 \;\Leftrightarrow\; q = \frac{c + v(l) + 1}{2}.$$

The memory complexity is thus determined by the maximum of 2^q and $2^{c(l+1)+1}$. It can be finally expressed as

$$\min_{0 \le l < T} \max(2^{\frac{c+v(l)+1}{2}}, 2^{c(l+1)+1}). \quad (2)$$

The plot of the memory complexity of the attack with structures compared to a standard differential attack is drawn in Figure 2. There c stands for the logarithm of the number of pairs required by the differential attack, q stands for the logarithm of the size of the structure that is used for the structural attack.

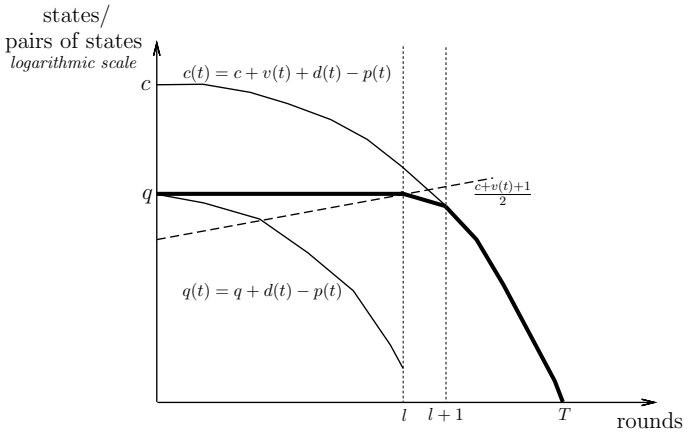

Fig. 2. Memory complexity of the collision search with structures.

4 Concrete attacks

4.1 How to construct a trail

The trails used in our attacks on Grindahl and Fugue have been obtained by a simple backtracking process. The idea is to start with zero-difference state and step back with introducing differences by all message injections. The differences spread to the internal state till every byte (or another building block) contains the difference. The number of steps is subject to the diffusion properties of the internal transformations.

4.2 Grindahl-256

Description. Grindahl is a family of hash functions proposed by Knudsen, Rechberger and Thomsen at FSE 2007 [13] as a stream-based hash function. The round function of Grindahl uses the design components of AES [7]: SubBytes and the MixColumns operation. Since the internal state of Grindahl is wider than that of AES (Grindahl-256 can be viewed as a byte matrix of 4 rows and 13 columns) it uses a modified ShiftRows transformation in order to obtain better diffusion. The other message-independent transformation is AddConstant, which adds a constant to a particular byte.

In Grindahl-256 the message injection is just the overwriting of the first column with 4-byte message block. The round function is defined as the following composition of transformations:

$$P(\alpha, M) = \textsc{MixColumns} \circ \textsc{ShiftRows} \circ \textsc{SubBytes} \circ \\ \circ \textsc{AddConstant} \circ \textsc{InjectMessage}(\alpha, M).$$

Here α denotes the state to be iterated, and M the message block to be injected. Every message block is used only once.

In order to obtain a hash value the state filled with zeros is iterated till the message is ended. Then eight blank rounds (no message injection) are applied and the resulting state is truncated to 256 bits, which is the hash value.

Security. The designers of GRINDAHL-256 claimed the security level of 2^{128} operations against both collision and second-preimage attack. Peyrin in [17] found a differential trail, which leads to a full collision in an internal state before the blank rounds are applied. The trail deals with two values of byte differences: non-zero and zero. It starts with a pair of states that differ in all bytes and after 9 message injections leads to a collision. Following our notation, he had 55 byte conditions, 21 byte degrees of value freedom, and 20 byte degrees of difference freedom thus obtaining complexity $2^{(55-21-20)\cdot 8} = 2^{112}$ message pairs. In early steps there was more freedom that is required by the trail so there was no clear difference between the value freedom and the difference freedom. However, the structural approach benefits from the difference freedom so we first exploit the latter one. There is also an attack on the prefix-MAC built on GRINDAHL [9].

Although GRINDAHL-256 is already broken, the goal of our attack is not only the illustration of structural technique. Peyrin provided some ad-hoc observations on the fact that his attack is one of the best dealing with the truncated-differential approach, and 2^{104} is the lower bound on the complexity of such attack. Our attack breaks this bound.

Attack. In order to apply the structural approach we first have to modify a bit the class of truncated differentials. Here and later we consider two-valued byte-difference: $*$ (random difference, including 0) and 0 (bytes coincide). They are marked as grey and white cells in Figure 3, respectively. One can easily check that this not only barely affect the probability of the trail and the complexity of the collision search but also simplify computations.

The second barrier is that the trail used by Peyrin for collision search is badly suited for the structural approach due to the distribution of probabilities among the iterations, which helps the standard differential attack but does not provide the best results for the structural attack. Table 1 (a) shows that we would have to start with a structure if $2^{12.5\cdot 8} = 2^{100}$ states, which does not offer enough advantage against Peyrin's attack.

The better complexity is provided by the second trail from [17], which was proposed for the second-preimage search. However, there is a mistake in Peyrin's paper: the byte C inserted before the k-th iteration does not affect column 11 in the k+1-th iteration. As a result, the complexity of a simple truncated differential attack is $2^{21\cdot 8} = 2^{168}$ pairs. However, the structural attack needs only a set of $2^{10.5\cdot 8+0.5} = 2^{84.5}$ states (Table 1 (b)).

The attack works as follows. Iterate GRINDAHL-256 for 10 rounds with randomly chosen messages and obtain a structure with $2^{84.5}$ states. Then we keep the size of the strcture after the first iteration thanks to the 4-byte difference

freedom. After the second round the structure collapses to 2^{72} pairs, and only one pair comes out of the next iteration.

i	v_i	d_i	p_i	$c(i)$	$\frac{c+v(i)}{2}$	$q(i)$
Start	–	–	–	14	7	12.5
1	0	2	2	14	7	12.5
2	3	4	7	14	8.5	9.5
3	4	3	7	14	10.5	5.5
4	4	2	7	13	**12.5**	1.5
5	4	3	9	11	14.5	–
6	4	4	14	5	16.5	–
7	2	2	9	0	17.5	–
$8-9$	0	0	0	0	17.5	–

(a)

i	v_i	d_i	p_i	$c(i)$	$\frac{c+v(i)}{2}$	$q(i)$
Start	–	–	–	21	10.5	10.5
1	0	4	4	21	**10.5**	10.5
2	4	4	20	9	12	–
3	4	3	15	0	14	–
$4-5$	0	0	0	0	14	–

(b)

Table 1. Parameters of differential trails for Grindahl-256. Measurement in bytes.

Time complexity of the attack. Since some message bytes pass several SubBytes transformations it is not clear how costly the steps when we deal with structures are. A trivial upper bound is $2^{q+\max(d_i)} \approx 2^{116}$. We propose some optimizations, which lead to a complexity about 2^{100} operations though the technique can probably be improved. The reader may refer to Table 8 for better understanding.

In the first step there are 4 bytes of difference freedom and 4 bytes where the difference should be canceled. The leader state of a new structure is defined by iterating the round function with a random message block. For each next state S in the structure we must find the message bytes (A, B, C, D) to be injected (we keep this notation in the further text) that lead to a state colliding in particular 4 bytes with the leader state. First consider column 7 before the MixColumns transformation in the second iteration. Three bytes of column 7 are not affected by the message injection and can be derived explicitly. On the other hand, one byte after the MixColumns transformation is known because a collision there is needed. Thus, compute both the input and the output of the MixColumns transformation of column 7 and thus derive the value of D and the value of second byte in column 9 in the next iteration.

Then try all the values of C. For each value derive one more byte in column 9 in the third iteration. As a result, two bytes in column 9 are known before the MixColumns transformation and two bytes are known due to the fact of collision. As a result, derive the values of A and B and check the MixColumns transformation in column 3 of the second iteration with the latter two values. On average, 2^7 trials of C are required.

The second step is actually the bottleneck of our attack, though we believe that the complexity may be reduced. First vary B and C for each state thus obtaining $2^{100.5}$ states. Then the 16 bytes in the third iteration where zero difference is desired are fully determined by 6 bytes that are affected by A and D. This gives us $16 - 6 = 10$ byte conditions, which can be used to divide the set of states into structures. One more condition we get from the second iteration, where the byte was affected by just fixed B. Therefore, we obtain $2^{(10+1)\cdot 8} = 2^{88}$ blocks each of size $2^{12.5}$. In every block we have 6 variables and 6 conditions; the other conditions are provided by constants. Since we process the blocks independently, the memory complexity is not increased.

Then consider the unknown bytes in columns 3, 5 and 11 that are affected only by A. Consider two random states in a block and denote by x_A and x'_A the message byte A after the SubBytes transformation. Then the fact of zero difference in those columns can be expressed as the following system of equations:

$$\begin{cases} a_{12}S(a_{21}x_A + c_1) + c_2 = a_{12}S(a_{21}x'_A + c'_1) + c'_2; \\ a_{11}S(a_{31}x_A + c_3) + c_4 = a_{11}S(a_{31}x'_A + c'_3) + c'_4; \\ a_{14}S(a_{41}x_A + c_5) + c_6 = a_{14}S(a_{41}x'_A + c'_5) + c'_6. \end{cases}$$

Here a_{ij} are coefficients of the MixColumns matrix and c_i are state-dependent constants. Due to properties of the AES S-box x_A and x'_A are uniquely determined (if there is a solution) by constants $\overline{c} = (c_1, c_2, c_3, c_4, c_5, c_6)$ and $\overline{c}' = (c'_1, c'_2, c'_3, c'_4, c_5, c_6)$. Furthermore, this property is transitive, so that we precompute the function $f : \overline{c} \to x_A$.

As a result, a block of 2^{12} states splits to 2^8 blocks with 2^4 states each where A, B and C are fixed. In order to obtain the value of D repeat the same trick in columns 7, 11, and 12 thus getting one pair per 2^4 states, or 2^{96} pairs at all. Only 2^{72} pairs of them pass through 3 conditions in column 9 in the fourth iteration.

In the last third step we have to pass 15 byte conditions given 6 byte degrees of freedom. Since we deal with separate pairs, the filtering process be maintained with precomputations (see [17]).

4.3 Grindahl-512

The hash function Grindahl-512 is defined similarly to Grindahl-256, but the internal state is twice as big as that of Grindahl-512: it has 8 rows and 13 columns. Each injection of a message block substitutes the first column with 8 bytes of a message. The row offset values are defined by the following expression:

$$c_i = i + 1;\ 0 \le i \le 7.$$

The MixColumns matrix is also redefined but the exact coefficients are irrelevant to our attack. The only property we use is that this matrix is MDS with branch number 9.

So far there is no collision attack on Grindahl-512 though a weakness of using Grindahl-512 as the base of prefix-MAC was shown [9].

Attack. We use a 3-round differential trail, which is shown at Figure 4. The trail is obtained by iterating GRINDAHL-512 backwards from the zero-difference state. It is assumed that the last truncation (before the injection) deletes a column with 6 byte differences, while the first two truncations delete the full-difference column. The parameters of the trail are listed in Table 2 at the left. However, the second step becomes so time-consuming that the resulting complexity overcomes the brute-force one. The reason is that structures are too large to be quickly recomposed into pairs. On the other hand, if we test all the possible injections, the time complexity increases as well.

We choose to decompose the second round into two sub-rounds with only slight increase of the complexity. The idea is as follows. We first process the zeros that are the result of the second MixColumns transformation and that are affected by the second message injection. These are 21 zeros in columns 1–8. For any two states that follow the trail before the second injection the condition of having zero difference in these positions is equivalent to 21 linear equations with the differences in the internal state after the S-box application as variables. Since the message injection can be equivalently swapped with the S-box transformation we obtain that the 21 equations are 21 linear conditions on 8 differences in the message block.

Therefore, $2^{13 \cdot 8}$ structures of size $2^{15 \cdot 8}$ split into $2^{(13+21-8) \cdot 8} = 2^{26 \cdot 8}$ structures of size $2^{2 \cdot 8}$. These structures collapse to pairs and are partly filtered out due to the remaining 28 byte conditions though 8 byte degrees of freedom are still available. Then we compose all possible $2^{30 \cdot 8}$ pairs and filter them out. The desired values to be injected can be derived from pre-computed tables, which are applicable since we already deal with pairs. The resulting complexity is 2^{240} computations and still 2^{224} memory. The complexity of the last step is negligible. We also modify the memory complexity table taking into account the considerations discussed above (Table 2).

i	v_i	d_i	p_i	$c(i)$	$q(i)$
	–	–	–	48	**28**
1	8	8	21	43	15
2	8	8	49	**10**	–
3	2	2	14	**0**	–

i	v_i	d_i	p_i	$c(i)$	$q(i)$
	–	–	–	48	**28**
1	8	8	21	43	15
2 - I	0	8	21	36	2
2 - II	8	0	28	**10**	–
3	2	2	14	**0**	–

Table 2. Parameters of the differential trail for GRINDAHL-512. The second table is obtained by splitting the second step into two substeps. Measurement in bytes.

4.4 Fugue

Hash family FUGUE [10] has been recently submitted to the SHA-3 contest [16], and has been recently chosen to the second round. It was designed by a group of

researchers in IBM. The design of FUGUE resembles that of GRINDAHL with several improvements, that should have increased the security. However, FUGUE is slower than GRINDAHL, which can be a serious disadvantage during the competition.

We analyze FUGUE with the structural approach and show that its security is much higher than that of GRINDAHL. Though we do not break the FUGUE security claims, the our attack is significantly faster than a trivial internal-collision attack.

Description

FUGUE-*256.* FUGUE-256 has internal state, denoted by S, of 120 bytes, which is viewed as a 4×30 array. We denote by S_i ($i = 0 \dots 29$) the i-th column of S. A message, appropriately padded, is split to 4-byte blocks. Each block I is an input to the round transformation of S, which is defined in pseudo-code as follows:

- TIX(I);
- Repeat 2 times:
  - ROR3;
  - CMIX;
  - SMIX;

TIX, ROR3 and CMIX are linear transformations. TIX consists of the following steps:

$$S_{16} += S_0; \quad S_0 = I; \quad S_8 += S_0; \quad S_1 += S_{24},$$

where + stands for XOR. CMIX is linear as well:

$$S_0 += S4; \quad S_1 += S_5; \quad S_2 += S_6; \quad S_{15} += S_4; \quad S_{16} += S_5; \quad S_{17} += S_6.$$

ROR3 rotates the state three columns to the right.

SMIX is a more complicated transformation. It process bytes in columns S_0–S_3. First, the AES S-box is applied to those 16 bytes. Then they are composed into a 16-byte vector, that is multiplied by matrix N, which is an almost-MDS matrix with branch number 16.

After all the blocks have been processed, the final round transformation is applied, and then eight columns of S are taken as hash output. Since we produce a collision before the final round, we skip its description (see full details in [10]).

FUGUE-*512.* FUGUE-512 follows the same philosophy, but has a stronger design: 36 columns (instead of 30) and twice as many operations as FUGUE-256 per round:

- TIX'(I);
- Repeat 4 times:
  - ROR3;

- CMIX';
- SMIX;

The CMIX' and TIX' operations have more column additions compared to FUGUE-256, and column indices are different. TIX':

$$S_{22} += S_0; \quad S_0 = I; \quad S_8 += S_0; \quad S_1 += S_{24}; \quad S_4 += S_{27}; S_7 += S_{30}.$$

CMIX':

$$S_0 += S_4; \quad S_1 += S_5; \quad S_2 += S_6; \quad S_{18} += S_4; \quad S_{19} += S_5; \quad S_{20} += S_6.$$

Properties of internal transformations. We consider truncated differentials, where difference in one byte may be either zero or random. We assume that two columns have equal differences with probability 2^{-32}, so every column addition in CMIX and TIX operations costs us 2^{32} if producing a zero column from two random ones. The SMIX transformation is more complicated. The matrix N is not MDS but is so called *almost MDS* with the branch number equal to 13. As a result, when constructing a trail in the backward direction, we get no benefit from having few active S-boxes in the input of S-Mix so we always assumed that any active S-Mix output was produced by the input where all the 16 bytes are active. We certainly assume that this approach may not be optimal though we do not see any properties of the S-Mix transformation which may lead to other possibilities.

Column	0–6	7–12	13	14–17	18–23	24–29
Depend on	-1	-2	-3	-1	-2	-3

Column	0–12	13–17	18–26	27–35
Depend on	-1	-2	-1	-2

Table 3. Column dependencies in FUGUE-256 and FUGUE-512. Value $-i$ for column j means that before r-th round the last message block that affected column j is M_{r-i}.

The designers provide several arguments for the resistance of FUGUE to pure and truncated differential attacks and even provide lower bounds for several attack modes, which unfortunately do not cover the mode that we use. We only point out that the complexity of the trivial internal collision attack on FUGUE is about $2^{29 \cdot 8 \cdot 2} = 2^{464}$ for FUGUE-256 and 2^{560} for FUGUE-512.

Analysis of Fugue-256. The optimal trail that we found for FUGUE-256 is a 6-round trail depicted in Table 5. Although differences in round $r + 2$ can theoretically be managed with a message injection in round r, this is not the case for this trail. We use the r-th message injection to get proper differences in only rounds r and $r + 1$ (mostly in round r).

We start with a structure of internal states of size $2^{44 \cdot 8} = 2^{352}$. It splits into 2^{320} structures of 2^{32} states each after three rounds (Table 4). About $2^{24 \cdot 8} = 2^{192}$

pairs come out of the next round, and we get one colliding pair after two more rounds. Due to big memory complexity of the attack, we assume that we are allowed to run much precomutation and store the results in tables. We thus assume that we spend negligible time complexity per each state and each pair, so the resulting time complexity should be about 2^{352} as well. This complexity is clearly much larger than the birthday bound (2^{128}) though it is at the same time much smaller than a birthday bound for the internal collision (2^{448}). We would also like to point out that we have not found any non-trivial differential attack with a comparable complexity.

i	v_i	d_i	p_i	$c(i)$	$q(i)$
Start	–	–	–	80	**44**
-6	0	4	4	80	44
-5	4	4	16	72	32
-4	4	4	32	48	4
-3	4	4	32	24	–
-2	4	4	32	0	–
-1	0	4	4	0	–

i	v_i	d_i	p_i	$c(i)$	$q(i)$
Start	–	–	–	116	**60**
-4	4	4	28	96	36
-3	4	4	56	48	–
-2	4	4	56	0	–
-1	0	4	4	0	–

Table 4. Parameters of differential trails for FUGUE-256 and FUGUE-512

Analysis of Fugue-512. The optimal trail that we found for FUGUE-512 is a 5-round trail depicted in Table 7. Here we use a message injection to get proper differences in the same round. We start with a structure of internal states of size $2^{60 \cdot 8} = 2^{480}$. It splits into 2^{192} structures of 2^{288} states in the next round (Table 4), and collapse to 2^{352} pairs after two rounds. Following the same observation, we again assume that we spend negligible time complexity per each state and each pair, so the resulting time complexity should be about 2^{480}, which is still much larger than the birthday bound (2^{256}) and smaller than a birthday bound for the internal collision (2^{560}).

5 Conclusions and future work

We showed how the organization of internal states into structures can drastically reduce the complexity of collision search providing an appropriate differential trail. The exact formulas for memory complexity and estimates on time complexity of the attack with structures have been provided. We successfully combined our approach with simply obtained differential trails and presented the best known attacks on GRINDAHL and the only external analysis of FUGUE. The results are summarized in Table 6.

R\C	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
−5	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
−4	*	*	*	*	*	−	−	*	*	*	*	−	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
−3	*	*	*	−	*	−	−	−	*	*	*	−	−	*	−	−	−	−	*	*	*	*	*	*	*	*	*	*	*	*
−2	*	*	−	−	*	−	−	−	*	−	*	−	−	−	−	−	−	−	−	−	−	−	−	−	*	*	*	*	*	*
−1	*	−	−	−	−	−	−	−	−	−	*	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
0	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−

Table 5. Trail for Fugue-256

We conclude that Fugue is much more resistant to attacks with truncated differentials, that were successfully used for the cryptanalysis of Grindahl. This is mostly due to a better diffusion and a larger internal state, which prevents from this style of attacks. We believe that our attacks can be further improved with other differential trails or better optimization of the maintenance of structures. The complexity of the attack is now determined by the bottleneck step when structures collapse to pairs. It is likely that the plot of the complexity function can be significantly smoothed for some hash functions.

Hash function	Attack	Memory complexity	Time complexity
Grindahl-256	Truncated differential [17]	2^{32}	2^{112}
	Structural	2^{84}	2^{100}
Grindahl-512	**Structural**	2^{224}	2^{240}
Fugue-256	Internal collision	2^{464}	-
	Structural	2^{352}	2^{352}
Fugue-512	Internal collision	2^{560}	-
	Structural	2^{480}	2^{480}

Table 6. Summary of our attacks on concrete hash functions.

Acknowledgements. The author greatly thanks anonymous reviewers for their valuable comments, which helped to improve the paper. The author is supported by PRP "Security & Trust" grant of the University of Luxembourg.

References

1. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Radiogatun, a belt-and-mill hash function, 2006, available at http://radiogatun.noekeon.org/.

2. Eli Biham. New techniques for cryptanalysis of hash functions and improved attacks on Snefru. In *FSE'08*, volume 5086 of *Lecture Notes in Computer Science*, pages 444–461. Springer, 2008.
3. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In *CRYPTO'90*, volume 537 of *LNCS*, pages 2–21. Springer, 1991.
4. Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.
5. Christophe De Cannière and Christian Rechberger. Finding SHA-1 characteristics: General results and applications. In *ASIACRYPT'06*, volume 4284 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.
6. Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher square. In *FSE'97*, volume 1267 of *LNCS*, pages 149–165. Springer, 1997.
7. Joan Daemen and Vincent Rijmen. *The Design of Rijndael. AES — the Advanced Encryption Standard*. Springer, 2002.
8. Ivan Damgård. A design principle for hash functions. In *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427. Springer, 1989.
9. Michael Gorski, Stefan Lucks, and Thomas Peyrin. Slide attacks on a class of hash functions. In *ASIACRYPT'08*, volume 5350 of *Lecture Notes in Computer Science*, pages 143–160. Springer, 2008.
10. Shai Halevi, William E. Hall, and Charanjit S. Jutla. The hash function fugue. Submission to NIST, 2008.
11. Antoine Joux. Multicollisions in iterated hash functions. Application to cascaded constructions. In *CRYPTO'04*, volume 3152 of *LNCS*, pages 306–316. Springer, 2004.
12. John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 474–490. Springer, 2005.
13. Lars R. Knudsen, Christian Rechberger, and Søren S. Thomsen. The Grindahl hash functions. In *FSE'07*, volume 4593 of *LNCS*, pages 39–57. Springer, 2007.
14. Stéphane Manuel and Thomas Peyrin. Collisions on sha-0 in one hour. In *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 2008.
15. Ralph C. Merkle. One way hash functions and DES. In *CRYPTO'89*, volume 435 of *LNCS*, pages 428–446. Springer, 1989.
16. NIST. Cryptographic hash algorithm competition. `http://www.nist.gov/hash-competition`.
17. Thomas Peyrin. Cryptanalysis of Grindahl. In *ASIACRYPT'07*, volume 4833 of *LNCS*, pages 551–567. Springer, 2007.
18. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO'93*, volume 558 of *LNCS*, pages 368–378. Springer, 1993.
19. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *EUROCRYPT'05*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

A Analysis of complexity

The time complexity analysis is much harder because we have to arrange states into structures as fast as possible.

We consider only one structure, because this process is independently applied for all structures. To obtain the whole complexity one should multiply the derived values by the current number of structures.

R\C	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26–30	31–35
-4	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	* * * * *	* * * * *
-3	*	*	-	-	*	-	*	*	*	*	*	-	*	*	*	*	*	*	*	*	*	*	*	*	*	*	* * * * *	* * * * *
-2	*	*	-	-	*	-	-	*	*	-	*	-	*	-	-	-	-	-	-	-	-	-	-	-	*	*	* * * * *	* * * * *
-1	*	-	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-----	-----
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-----	-----

Table 7. Optimal trail for FUGUE-512

No freedom in message injection. States S'_i and S'_j belong to the same structure if $S'_i \sim S'_j$. On the other hand, if $\sim$ defines set R of linear differences then the condition can be expressed in terms of projections to space $R^\perp$ that is orthogonal to R:

$$S'_i \sim S'_j \;\Leftrightarrow\; \mathrm{pr}_{R^\perp} S'_i = \mathrm{pr}_{R^\perp} S'_j.$$

As a result, we compute the ordered set of projections of structures and use binary search to derive the structure a state belongs to. Assuming the sorting and search costs are negligible comparing to the round iteration we derive the complexity roughly equal to the number of states.

Value freedom in message injection. The fact that there is the value freedom in message injection does not affect the complexity of the attack if structures do not collapse into pairs yet. The exact value of the complexity in this case is just the number of states after the round iteration.

Consider the case where structures are collapsed to pairs and single states. As mentioned in Section 3, the structure contains 2^{2q-1+v} pairs. The further steps depend on whether we can exploit properties of the round function.

- If the round function is viewed as a black box, we just derive pairs for each possible message m. The complexity is about 2^{q+v}.
- If we can quickly find solutions m for the equation

$$\mathrm{pr}_{R^\perp} S_i[m] = \mathrm{pr}_{R^\perp} S_j[m] \tag{3}$$

  then it is solved for all possible pairs about 2^{2q-1} times.
- If there exist not only a fast algorithm for solution (3) but also function f such that (3) has a solution iff $f(S_i) = f(S_j)$. Then we compose the ordered set of $f(S)$ and for each new state look for a pair with negligible cost. The complexity would be equal to the maximum of the size of the initial structure (2^q) and the number of resulting pairs ($2^{2q-1+v-p}$).

Difference freedom in message injection. Again, first, we investigate the case where structures do not collapse to pairs. Suppose states $S'_1, \ldots, S'_i$ have been

already distributed into just created structures. We also require that every leader state is obtained by the same injected message m_0. A state S_{i+1} can be distributed to the structure with the leader state S' if there exist a message m_{i+1} such that $m_0 \sim m_{i+1}$ and $\mathrm{pr}_{R^\perp} S' = \mathrm{pr}_{R^\perp} S_{i+1}[m_{i+1}]$. Denote by S the set of all such states $S_{i+1}[m_{i+1}]$. Then the question is whether $\mathrm{pr}_{R^\perp} S'$ belongs to $\mathrm{pr}_{R^\perp} \mathsf{S}$.

If S is an affine space, and the linear space does not depend on S_{i+1} then we can easily compute the projection and find the corresponding structure using the ordered set approach. The complexity would be equal to the number of states. If S is not an affine space but can be represented as a union of affine spaces then we compute the projection for each space. In the worst case the complexity is equal to 2^d multiplied by the number of states.

Now consider the case where a structure collapse to pairs. This is actually the most complicated case and can be considered as a bottleneck. Indeed, about $2^{2q-1+v+d}$ pairs are composed from a structure with 2^q states. About $2^{2q-1+v+d-p}$ pairs come out of the round iteration. The possible approaches are similar to the case where there is no freedom in difference. If the round function is a black box, the complexity varies from 2^{q+v} to 2^{q+v+d}. If there exists a function f such that (3) has a solution iff $f(S_i) = f(S_j)$, then the complexity is between 2^q and $2^{2q-1+v+d-p}$).

B Trails

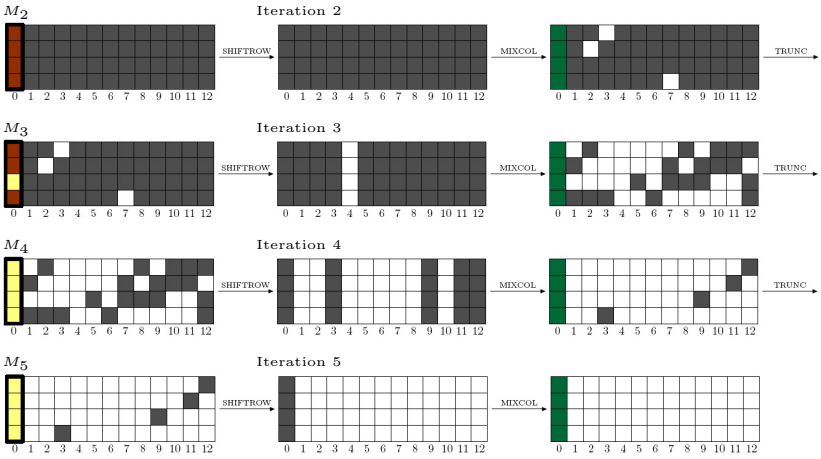

Fig. 3. Differential trail for GRINDAHL-256 (Table 1 (b)).

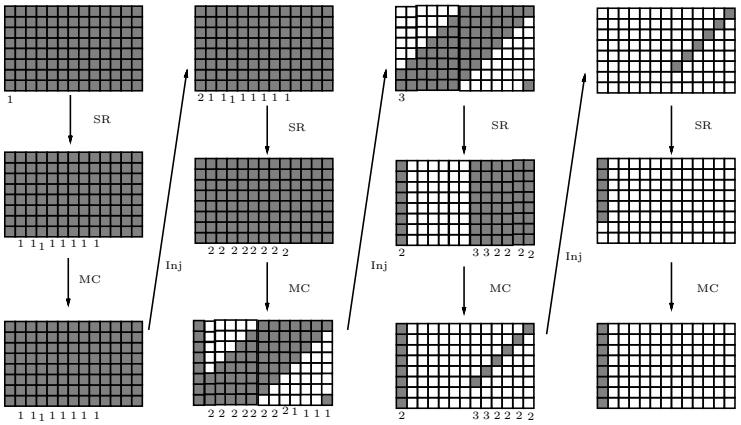

Fig. 4. Differential trail for GRINDAHL-512 (Table 2 (b)).

			Message bytes											
			1				2				3			
It	Col	Cost												
2	2	1						B						
	3	1	B	A										
	7	1				D								
3	1	2									A			
	2	2										B		
	3	3					B	A						
	5	3					C		A					
	6	3						C	B					
	7	2								D				
	8	2							C					
	9	2	C	D	AC	BD								
	10	2												D
	11	2					D			A				
	12	1						D		B				
4	3	3									B	A		
	9	3					C	D	AC	BD				
	11	3									D			A
	12	3										B		

Table 8. Dependencies of the message block in the differential trail for GRINDAHL-256.