Integral Distinguishers of Some SHA-3 Candidates

Marine Minier[1], Raphael C.-W. Phan[2], and Benjamin Pousse[3]

[1] Universit de Lyon, INRIA
INSA-Lyon, CITI, F-69621, Villeurbanne, France
marine.minier@insa-lyon.fr
[2] Electronic and Electrical Engineering, Loughborough University
LE11 3TU Leicestershire - UK
R.Phan@lboro.ac.uk
[3] XLIM (UMR CNRS 6172), Université de Limoges
23 avenue Albert Thomas, F-87060 Limoges Cedex - France
benjamin.pousse@unilim.fr

Abstract. In this paper, we study structural Integral properties on reduced versions of the compression functions of some SHA-3 candidates: Hamsi-256, LANE-256 and Grøstl-512. More precisely, we improve on the Integral distinguishers of Hamsi-256 (less time complexity or deterministic instead of probabilistic) and present the first known Integral distinguishers for LANE-256 and improved Integral distinguisher for Groestl-512. Whereas the SHA-3 competition focuses the cryptographic world attention on the design and the attacks of hash functions, results in this paper analyze the resistance of some SHA-3 candidates against structural properties built on Integral distinguishers.

Keywords: hash functions, cryptanalysis, integral distinguishers, SHA-3 candidates.

1 Introduction

Today, with the cryptographic world focus on the SHA-3 competition[1], recent cryptanalytic attacks are mounted across both hash function primitives and underlying ciphers/permutations. Notably, the joint analysis of hash functions and underlying block ciphers or permutations has led to considerations of new attack models for block ciphers, e.g. known key [20] or chosen key [5]. In fact, when block cipher inspired permutation structures are used as building blocks within hash functions, there is no secret (key) input, thus the building block is not only a known transformation, it is also a computable one. Interestingly enough, doing so has led to the discovery of more powerful new techniques to construct distinguishers and/or mount key recovery attacks for block ciphers back in the conventional unknown key model where ciphers are standalone constructs instead

[1] http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

S.-H. Heng, R.N. Wright, and B.-M. Goi (Eds.): CANS 2010, LNCS 6467, pp. 106–123, 2010.

of underlying hash functions, including the first known related-key attacks on the full version of AES-256 and AES-192 [5,4].

In this paper, we mainly focus on Integral properties and their applications in the known transformation model to find structural properties of some of the SHA-3 candidates: Hamsi-256, LANE-256 and Grøstl-512. As shown in [25,2], particular Integral properties essentially depending on the structure of the linear part of the cipher could be exhibited and are powerful tool for cryptanalytic studies. Those integral properties starting from the middle of the cipher lead to distinguishers in the known key settings defined in [20] and for the underlying compression functions of hash functions. In more details, our contributions are as follows. For Hamsi-256, we present improved Integral distinguishers for its full compression function: for its P_f permutation, our distinguisher requires less time complexity, for its P permutation our distinguisher is deterministic versus the previous known distinguisher that is probabilistic. For LANE-256, we present the first known Integral distinguishers. For Groestl-512, we present an Integral distinguisher that is improved over the best previously known Integral distinguisher.

This paper is organized as follows: Section 2 introduces the related work in the field of integral cryptanalysis and the notations that will be used in the rest of this paper. Sections 3, 4 and 5 respectively present integral properties and distinguishers on the Hamsi-256, LANE-256 and Grøstl-512 compression functions. Finally, Section 6 concludes this paper.

2 Related Work and Notations

Integral cryptanalysis was first introduced by Knudsen against the block cipher Square in the original paper [10] in the unknown key model to retrieve information on some key bytes. Then, it was applied to the AES in the original submission paper [11,12] and later the distinguisher was extended by one round by Ferguson et al. in [14].

After those first attacks, many ciphers especially the ones that use an SPN structure have been studied in regard of this kind of distinguishers. Among the integral cryptanalyses proposed in the literature, we could cite the attacks against SAFER [3], CRYPTON [13] and more recently on PRESENT [9]. The different Rijndael versions (Rijndael-192 and Rijndael-256) have also been attacked using integral properties [19,15]. Other contributions also analyze the general framework of Integral cryptanalysis and especially focus on the required conditions for a block cipher to be attacked using this method [21,6]. In [21], Knudsen and Wagner analyze integral cryptanalysis as a dual to differential attacks notably applicable to block ciphers with bijective components. A first-order integral cryptanalysis considers a particular collection of m words in the plaintexts and ciphertexts that differ on a particular word. The aim of this attack is thus to predict the values in the sums (i.e. the integral) of the chosen words after a certain number of rounds of encryption. The same authors also generalize this approach to higher-order integrals: the original set to consider

becomes a set of m^d vectors which differ in d words and where the sum of this set is predictable after a certain number of rounds. The sum of this set is called a dth-order integral.

More recently, in [20] Integral cryptanalysis has been proposed in the new cryptanalysis model called known key setting where the key is known to the attacker. In the same setting, compression functions of hash functions could also be analyzed and some distinguishers have been proposed against SHA-3 candidates using Integral properties. As example we could cite integral distinguishers on the compression functions of Hamsi-256 [2] and Keccak [7].

In the rest of this paper, we use the consistent notations introduced in [21] and extend them for expressing word-oriented integral attacks. For a dth order integral, we have:

- The symbol '$\mathcal{C}$' (for "Constant") in the ith entry, means that the values of all the ith words in the collection of texts are equal.
- The symbol '$\mathcal{P}$' (for "Permutation") means that all words in the collection of texts are different.
- The symbol '?' means that the sum of words cannot be predicted.
- The symbol '$\mathcal{P}^d$' corresponds with the components that participate in a dth-order integral, i.e. if a word can take m different values then $\mathcal{P}^d$ means that in the integral, the particular word takes all values exactly m^{d-1} times.
- The symbol '$\mathcal{B}$' (for "Balanced") means that the sum of all values is zero.
- The symbol 'Eq_i' (for "Equality') found for two different words means that the sums of all values taken on those particular words matched (i.e. are equals).

3 Hamsi-256

In this section, we will introduce new Integral properties first on the compression function P of Hamsi-256 that happens with probability 1 contrary to the one proposed in [2] and then on the final transformation P_f. Many recent results concerning Hamsi cryptanalysis have been presented in [8] and in [2]. As previously mentioned, some probabilistic integral properties have been presented in [2] and also an integral distinguisher on P_f which has a complexity of 2^{28} computations. The rest of the cryptanalytic results on Hamsi-256 essentially concerns differential paths that happen with low probabilities to build message-recovery attacks and pseudo-second-preimage attacks.

3.1 Description of the Hamsi-256 Hash Function

This section describes the hash function Hamsi-256. We refer to [22] for a complete specification.

General View. Hamsi is based on the Concatenate-Permute-Truncate design strategy. It uses in addition a message expansion and a feed forward of the chaining value in each iteration. Thus the compression function of Hamsi can be divided into four mappings:

- **Message Expansion:** $E : \{0,1\}^{32} \rightarrow \{0,1\}^{256}$
- **Concatenation:** $C : \{0,1\}^{256} \times \{0,1\}^{256} \rightarrow \{0,1\}^{512}$
- **Non linear permutations P and P_f:** $\{0,1\}^{512} \rightarrow \{0,1\}^{512}$
- **Truncation:** $T : \{0,1\}^{512} \rightarrow \{0,1\}^{256}$

The message M to hash is properly padded and cut into l 32-bit blocks $M_1, \cdots M_l$. Each block is transformed using the message expansion into a 256-bit block seen as a 4×4 matrix of 32-bit words. Once expanded, each block is processed by the compression function. Starting from a predefined initial value h_0, the compression function H (or H_f for the final transformation) could be written as follows to compute the digest h of M:

$$h_i = H(h_{i-1}, M_i) = (T \circ P \circ C(E(M_i), h_{i-1})) \oplus h_{i-1} \text{ for } 0 < i < l$$
$$h = H(h_{l-1}, M_l) = (T \circ P \circ C(E(M_l), h_{l-1})) \oplus h_{l-1}$$

Internal mappings

Message expansion. The message expansion of Hamsi-256 is based on a linear code given by its generator matrix G (see [22] for more details). Thus:

$$E(M_i) = (m_0, \cdots, m_7) = (M_i \times G)$$

where $(m_0, \cdots, m_7)$ are eight 32-bit words.

Concatenation. The concatenation function C builds a 512-bit word from the 256-bit expanded message $(m_0, \cdots, m_7)$ and the 256-bit chaining value $h_i = (c_0, \cdots, c_7)$ at 32-bit word level:

$$C(m_0, \cdots, m_7, c_0, \cdots, c_7) = (m_0, m_1, c_0, c_1, m_2, m_3, c_2, c_3, m_4, m_5, c_4, c_5, m_6, m_7, c_6, c_7)$$

corresponding, when looking at the matrix representation, with rows that are composed of two message words and of two chaining value words.

Truncation. The truncation function T selects eight 32-bit words among the 16 from the internal state to form the new chaining value after feed forward:

$$T(s_0, s_1, s_2, \cdots, s_{14}, s_{15}) = (s_0, s_1, s_2, s_3, s_8, s_9, s_{10}, s_{11})$$

corresponding, when looking at the matrix representation, with choosing the second and the fourth rows.

Non-linear permutations. P and P_f use the same size parameters (the input and output blocks are 512-bit long) and the same round function. They differ by the round constants and by the number of rounds: three rounds for P and six rounds for P_f (for this latter in fact 8 rounds are the recommended parameters of the designers even if the specifications are always provided with 6 rounds). The round function is composed of three layers:

- Addition of constants and counter: predefined constants are first XORed to the whole internal state. Then, a counter is XORed to s_1, the second 32-bit word of the internal state.
- Substitution layer: it uses the 4-bit Sbox of the block cipher Serpent [1] in a bitslice mode. From each of the four 32-bit words of a same column, the four bits at the same position are extracted and replaced by the corresponding value of the Sbox.
- Linear diffusion layer: it applies the Serpent linear transform L from $\{0,1\}^{128}$ into itself to each diagonal of the state matrix:

$$\begin{aligned}(s_0, s_5, s_{10}, s_{15}) &= L(s_0, s_5, s_{10}, s_{15})\\ (s_1, s_6, s_{11}, s_{12}) &= L(s_1, s_6, s_{11}, s_{12})\\ (s_2, s_7, s_8, s_{13}) &= L(s_2, s_7, s_8, s_{13})\\ (s_3, s_4, s_9, s_{14}) &= L(s_3, s_4, s_9, s_{14})\end{aligned}$$

The linear transformation L could be written in pseudo-code as follows for a four 32-bit words input (a, b, c, d):

$a := a <<< 13; \quad c := c <<< 3; \quad b := b \oplus a \oplus c; \quad d := d \oplus c \oplus (a << 3); \quad b := b <<< 1;$
$d := d <<< 7; \quad a := a \oplus b \oplus d; \quad c := c \oplus d \oplus (b << 7); \quad a := a <<< 5; \quad c := c <<< 22;$

where $<<<$ denotes left rotation and $<<$ denotes left shifting.

3.2 Integral Properties of Hamsi-256

We first give an Integral property on the permutation P that only depends on the chaining value h_i then we analyze an integral distinguisher on the permutation P_f. When looking in detail at the diffusion function of P, we could see that a complete diffusion could not be reached after three rounds. So, we have exhibited the integral property described in Fig. 1, which was tested and verified on 2×10^6 random values. This particular integral property leads to a distinguisher that uses $(2^2)^4 = 2^8$ values and that leads to values that sum to zero on 444 bits. In the same way when trying to limit the number of inputs, if the same four particular words take all possible values only on the least significant bit, this integral property leads to sums equal to 0 on about 30 bits after the P application.

When looking at an integral distinguisher for the final transformation P_f, we start from the middle as done when known key settings are considered. Thus, we are looking for integral properties in the forward and in the backward senses on three rounds. Fig. 2 presents the integral property found using intensive computations in the forward sense for three P_f rounds. Thus, we obtain a three forward rounds integral distinguisher that uses 2^{16} values that sum to zero everywhere.

In the same way, we have exhibited the three backward rounds integral property shown in Fig. 3. Thus, we obtain a three backward rounds integral distinguisher that uses 2^{16} values that sum to zero everywhere.

Thus, we could combine those two properties in the forward and in the backward senses starting from the middle using 2^{16} middletexts that are constant

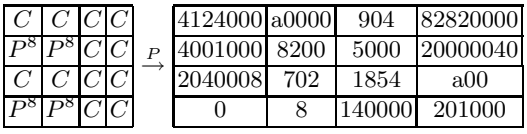

Fig. 1. The Integral property for P where P^8 means that the four considered 32-bit words are constant everywhere except on the 2 least significant bits that take all possible values (between 0 and 3) and where the values given in hexadecimal notation for the output corresponds with the output mask (i.e. a 0 value means that the sum on this byte is always equal to 0)

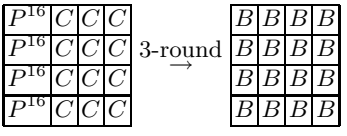

Fig. 2. The Integral property for 3 P_f rounds in the forward sense where P^{16} means that the four considered 32-bit words are constant everywhere except on the 4 least significant bits that take all possible values (between 0 and 15). All the outputs have the property $\mathcal{B}$ and sum to zero.

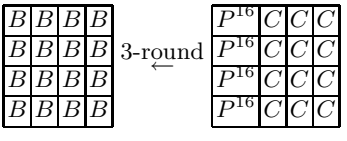

Fig. 3. The Integral property for 3 P_f rounds in the backward sense where P^{16} means that the four considered 32-bit words are constant everywhere except on the 4 least significant bits that take all possible values (between 0 and 15). All the outputs have the property $\mathcal{B}$ and sum to zero.

everywhere except on the 4 least significant bits of four particular words. From this set of texts, if we go forward, we obtain a set of values that sum to zero everywhere and if we go backward, we also obtain a set of values that sum to zero everywhere. Thus, we have exhibited a particular integral distinguisher that is such that starting from 2^{16} particular middletexts leads to input and output values that sum everywhere to 0. The computational cost for this distinguisher is 2^{16} calls to the transformation P_f.

4 LANE-256

4.1 Description of the LANE-256 Hash Function

The cryptographic hash function LANE [18] has been submitted to the NIST SHA-3 competition and has been discarded after round 1 due to the attack

presented in [23]. This attack against the whole version of LANE-256 presents semi-free-start collisions that could be constructed in 2^{96} computations using 2^{88} memory.

LANE-256 is an iterated hash function that supports four digest sizes (224, 256, 384 and 512 bits) and the use of a salt. We focus here on LANE-256 where the initial chaining value H_{-1} has a 256 bits size. The message is padded and split into message blocks M_i of length 512 bits for LANE-256. Then the compression function f of Lane-256 transforms iteratively 256 bits of the chaining value and 512 bits of the message block into a new chaining value of 256 bits $H_i = f(H_{i-1}, M_i, C_i)$ where C_i is a counter that indicates the number of message bits processed so far. Finally, after all the message blocks are processed, the final digest is derived from the last chaining value, the message length and the salt by an additional call to the compression function. For the detailed structure of the compression function we refer to the specification of LANE [18]. First, the chaining value and the message block are processed by a message expansion that produces an expanded state with doubled size. Then, this expanded state is processed in two layers. The first layer is composed of six permutations $P_0, \cdots, P_5$ applied in parallel, and the second layer of two parallel layers Q_0, Q_1.

The message expansion of LANE takes a message block M_i and a chaining value H_{i-1} and produces the input to six permutations $P_0, \cdots, P_5$. In LANE-256, the 512-bit message block M_i is split into four 128-bit blocks $m_0||m_1||m_2||m_3$ and the 256-bit chaining value H_{i-1} is split into two 128-bit words $h_0||h_1$. Then, six more 128-bit words $a_0, a_1, b_0, b_1, c_0, c_1$ are computed:

$$a_0 = h_0 \oplus m_0 \oplus m_1 \oplus m_2 \oplus m_3,\ a_1 = h_1 \oplus m_0 \oplus m_2,$$
$$b_0 = h_0 \oplus h_1 \oplus m_0 \oplus m_2 \oplus m_3,\ b_1 = h_0 \oplus m_1 \oplus m_2,$$
$$c_0 = h_0 \oplus h_1 \oplus m_0 \oplus m_1 \oplus m_2,\ c_1 = h_0 \oplus m_0 \oplus m_3.$$

Each of these 128-bit values, as in AES, can be seen as 4×4 matrix of bytes. In the following, we will use the notation $x[i, j]$ when we refer to the byte of the matrix x with row index i and column index j, starting from 0. The values $a_0||a_1, b_0||b_1, c_0||c_1, h_0||h_1, m_0||m_1, m_2||m_3$ become the inputs of the six permutations $P_0, \cdots, P_5$ described below.

Each permutation P_i operates on a state that can be seen as a double AES state ($2\times$ 128-bits). The permutation reuses the transformations SubBytes (SB), ShiftRows (SR) and MixColumns (MC) of the AES with the only exception, that due to the larger state size, they are applied twice in parallel. Additionally, there are three new round transformations introduced in LANE. AddConstant (AC) adds a different value to each column of the state and AddCounter (ACO) adds part of the counter C_i to the state. The third transformation is SwapColumns (SC) used for mixing parallel AES states. It swaps the two right columns of the left half-state with the two left columns of the right half-state.

The complete round transformation consists of the sequential application of all these transformations in the given order. The last round omits AddConstant and AddCounter. Each of the permutations P_i consists of six rounds for LANE-256.

The permutations Q_0 and Q_1 are composed of three previously defined rounds where the last round does not contain the AddConstant and AddCounter operations.

4.2 Integral Properties of LANE-256

As the diffusion layer of LANE-256 has a really slow diffusion, we could expect that the same kind of properties as the ones presented in [25] exist on 5 rounds or more. When studying in detail the integral properties, we obtain a 2nd order integral property on 4-round of each P_i as shown in Fig. 9 in Appendix A. This 4 forward rounds integral property could be extended by two rounds at the beginning using the two extensions (an 8th order and a 16th order integral respectively) shown in Fig. 11 in Appendix A. Thus, using 2^{128} chosen plaintexts we are able to distinguish the output after 6 forward rounds of all the P_i functions from a random permutation because the sums taken at byte level over all the inputs are equal to 0 for all the output bytes. The complexity of the distinguisher is equal to 2^{128} P_i operations.

Similarly in the backward sense, we found a 2nd order integral property on 3 backward rounds presented in Fig. 10 in Appendix A. This property leads to a distinguisher on 3 backward rounds where the sums taken at byte level over all the inputs are equal to 0. It requires 2^{16} chosen texts to work and has a complexity equal to 2^{16} operations. This property could be extended by one backward rounds at the beginning using an 8th order integral property as shown on Fig. 12 in Appendix A. This leads to an integral distinguisher that uses 2^{64} chosen plaintexts with a complexity equal to 2^{64} operations to test if the sums taken at byte level over the 256 bits are equal to 0 or not.

We could combine those two properties (in the backward and in the forward senses) starting from the middle of one particular P_i (say P_4) and the corresponding Q_i (say Q_1) to build a structural property on the right part of the LANE compression function (see Fig. 13 in Appendix A). For the permutation P_4, start in the middle (after 4 rounds) with 2^{112} middletexts with 14 active bytes (the other are taken equal to a constant) then, go backward on four rounds to obtain inputs that sum to 0 everywhere and go forward on 5 rounds to obtain outputs that sum to 0 everywhere.

In the same way, using the previous property on the right part of the compression function, we could obtain a complete property on the LANE-256 compression function when P_0, P_1 and P_2 are limited to 3 rounds (instead of 6 in the original version) using our 16th order integral property on the left part composed of 6 rounds when concatenating P_0 and Q_0. To do so, repeat the property on the right part considering that h_0 and h_1 are constants, for 2^{128} $h_0||h_1$ values that correspond to a 16th order integral. Thus, and as shown in Fig. 14 in Appendix A, we have exhibited an intrinsic property of the compression function of LANE when P_0 has only 3 rounds (instead of 6) using 2^{240} values seen as in the one hand 2^{128} copies of the 2^{112} middletexts that lead to integral properties on the right part of LANE-256 and on the other hand 2^{112} copies of a 16th order

integral 6-round property. The complexity required to exhibit this property is equal to 2^{240} P_i operations whereas the memory requirements are small.

Note also that the same kind of properties could be directly deduced for LANE-512.

5 Grøstl-512

5.1 Description of the Grøstl-512 Hash Function

Grøstl [16] is a SHA-3 candidate designed by Guaravaram *et al.*, notably Grøstl-256 outputs hash value of length 224 and 256 bits whereas Grøstl-512 outputs hash value of length 384 or 512 bits. We mainly focus here on Grøstl-512. It is an iterated hash function with a compression function built from two distinct permutations P and Q. A t-block message M (after padding) is hashed using the compression function $f(H_{i-1}, M_i)$ and output tranformation $g(H_t)$ as follows:

$$\begin{aligned} H_0 &= IV \\ H_i &= f(H_{i-1}, M_i) = H_{i-1} \oplus P(H_{i-1} \oplus M_i) \oplus Q(M_i) \text{ for } 1 \leq i \leq t \\ h &= g(H_t) = trunc(H_t \oplus P(H_t)) \end{aligned}$$

The two permutations P and Q are constructed using the wide trail strategy, their design is very similar to the AES with a fixed key input. Both permutations of Grøstl-512 act on a 1024-bit state represented as a 8×16 matrix of bytes and have 14 rounds. The round transformations of Grøstl-512 are the following ones:

- AddRoundConstant (AC) adds different one-byte round constants to the 8×16 states of P and Q.
- SubBytes (SB) is the non-linear layer that applies the AES Sbox to each byte of the state.
- ShiftBytes (ShB) rotates the bytes of row j in the following way: 0 for $j = 1$, 1 for $j = 2, \cdots$ 6 for $j = 7$ and 11 for $j = 8$.
- MixBytes (MB) is the linear diffusion layer where each column of the state is multiplied by a constant matrix B.

5.2 Grøstl Analysis So Far

Grøstl has attracted significant amount of cryptanalysis. Among the best cryptanalytic results on the two Grøstl version (Grøstl-256 and Grøstl-512), we could mention semi-free-start collisions on the compression function of Grøstl-256 reduced to seven rounds presented in [17] that have a complexity of 2^{120} computations and 2^{64} in memory; and semi-free-start collisions on the compression function of Grøstl-512 reduced to eight rounds presented in [24] that have a complexity of 2^{152} computations and 2^{64} in memory.

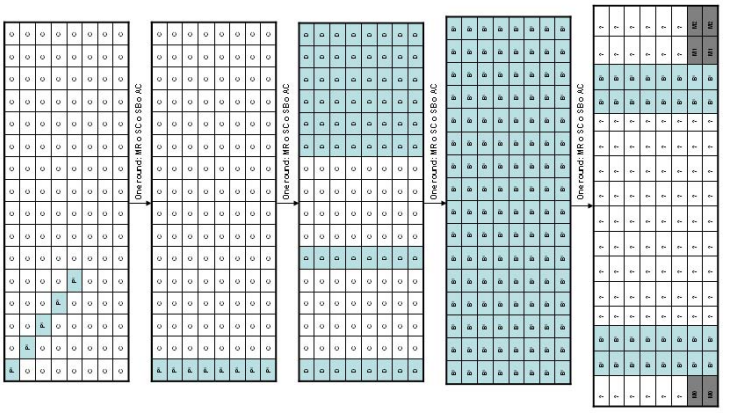

Fig. 4. The 5th order Integral property on 4 rounds of P and Q

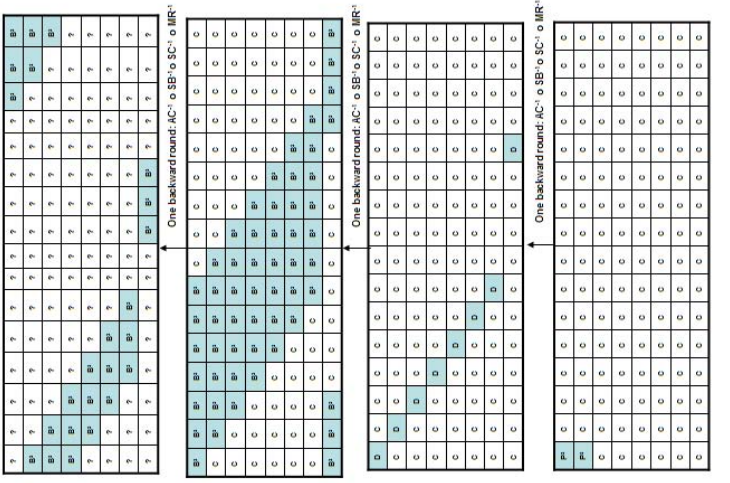

Fig. 5. The 2nd order Integral property on 3 backward rounds of P and Q

Concerning particular distinguishers, structural non-random properties have been observed e.g. fixed points [16] and q-multicollisions in the original submission; and distinguishers have been presented on Grøstl-256 reduced to 8 out of 10 rounds. In a more recent article [26], Thomas Peyrin presents distinguishers using truncated differential properties on the 10 rounds of the compression function of Grøstl-256 requiring 2^{192} computations and 2^{64} memory; and on 11 rounds of the compression function of Grøstl-512 requiring 2^{640} computations and 2^{64} memory.

5.3 Integral Properties of Grøstl-512

As the diffusion layer of Grøstl-512 has a really slow diffusion, we could expect that the same kind of properties as the ones presented in [25] may exist on 5 rounds or more. Studying in detail the diffusion layer, we obtain the 5th order integral distinguisher on 4 rounds of P and of Q shown in Fig. 4.

This distinguisher leads to the sums taken at byte level over all the inputs on the four columns marked in blue in Fig. 4 are equal to 0. Moreover, there are some particular matching (equality) sums (marked in gray in Fig. 4) between the bytes at positions $(1, 7)$ and $(1, 8)$ and also at positions $(15, 7)$ and $(15, 8)$ and positions $(16, 7)$ and $(16, 8)$. This property requires 2^{40} chosen texts to work and has a complexity equal to 2^{40} permutation operations.

This property could be extended by one round at the beginning using a 40th order integral property as shown on Fig. 6. This leads to an integral distinguisher that uses 2^{320} chosen plaintexts with a complexity equal to 2^{320} permutation operations to test if the sums taken at byte level over $8 \times 8 \times 4 = 256$ bits are equal to 0 or not. The sums taken on the equalities could also be considered leading to 0-sums on 230 bits.

Let us now analyze which are the integral properties that exist for the backward sense. We found the 2nd order integral property on 3 backward rounds presented in Fig. 5.

This property leads to a distinguisher on 3 backward rounds where the sums taken at byte level over all the inputs on the three shifted columns marked in blue in Fig. 5 are equal to 0. It requires 2^{16} chosen texts to work and has a complexity equal to 2^{16} cipher operations. This property could be extended by two backward rounds at the beginning using a 64th order integral property as shown on Fig. 5. This leads to an integral distinguisher that uses 2^{512} chosen plaintexts with a complexity equal to 2^{512} permutation operations to test if the sums taken at byte level over $3 \times 8 \times 4 = 192$ bits are equal to 0 or not.

We could combine those two properties (in the backward and in the forward senses) starting from both the middle of P and the middle of Q to build a structural property on the compression function of Grøstl-512 when 10 rounds are considered (see Fig. 8). For the permutation P, start from the middle with 2^{512} middletexts with 64 active bytes (the other are taken equal to a constant) then, go backward on five rounds to obtain inputs that sum to 0 on 3 shifted columns

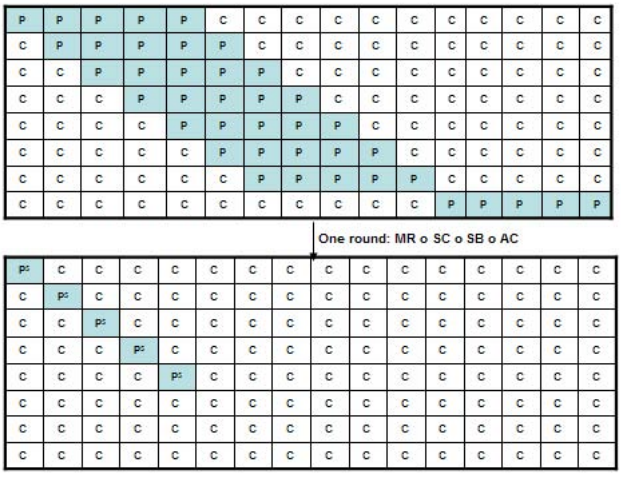

Fig. 6. Extension by one round of the previous distinguisher using a 40th order Integral property

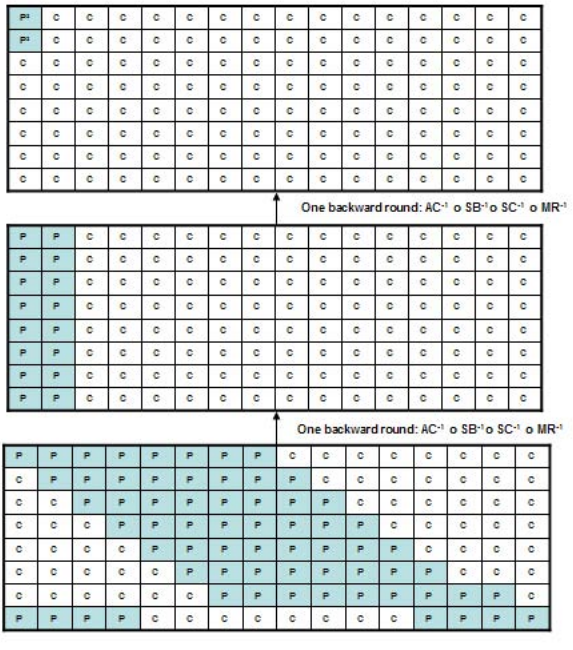

Fig. 7. Extension by two backward rounds of the 3 backward rounds distinguisher using a 64th order Integral property for P and of Q

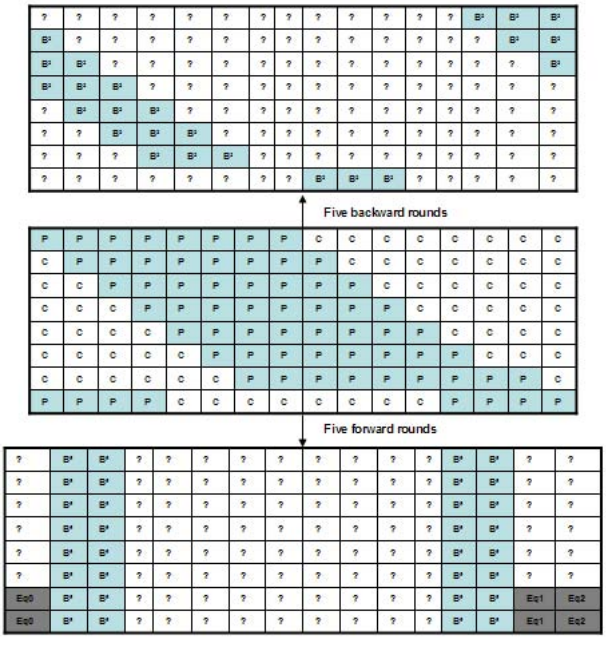

Fig. 8. Complete property on 10 rounds of P and Q starting from the middle with a 64th order integral property

and go forward on 5 rounds to obtain outputs that sum to 0 on 4 columns. Do the same for the permutation Q. Using Q, get the 2^{512} corresponding M_t messages. Using those messages and the inputs of P, compute the corresponding 2^{512} H_{t-1} values. Those 2^{512} values also verify that their sums taken over all the 2^{512} values on 3 shifted columns are equal to 0 (due to the linearity of the xor operation). Then with the knowledge of H_{t-1}, of the outputs of P and of the outputs of Q, the corresponding H_t values are such that the sums taken over all the 2^{512} values on the intersection of the 3 shifted columns (for the backward sense) and of the 4 columns (for the forward sense) are equal to 0. In other words, the sum taken over all the 2^{512} outputs of the compression function is zero at 7 byte positions whereas the corresponding inputs H_{t-1} and M_t have 0-sum on 3 shifted columns.

Thus, we have exhibited a structural property of the Grøstl-512 compression function when P and Q are limited to 10 rounds. The computational cost of this property is about 2^{513} operations with few memory requirements to find some 0-sums at particular positions (7 bytes at the output of the compression function and 24 bytes at the input). Note that this new structural property really improves the one described in the original proposal of Grøstl [16] that reaches 9 rounds with a complexity equal to 2^{704} cipher operations.

Table 1. Summary of distinguishers and of attacks on the three studied candidates

Hash functions	Nb rounds	Type of Attack	Time	Memory	Source
Hamsi-256	P_f	Integral Dist.	2^{28}	small	[2]
Hamsi-256	P	Prob. Int. Dist.	2^{2}	small	[2]
Hamsi-256	P_f	Integral Dist.	2^{16}	small	this paper
Hamsi-256	P	Integral Dist.	2^{8}	small	this paper
LANE-256	complete	Semi-free-start Coll.	2^{96}	2^{88}	[23]
LANE-256	(3,3,6,3)	Integral Dist.	2^{240}	small	this paper
Grøstl-512	8	Semi-free-start Coll.	2^{152}	2^{64}	[24]
Grøstl-512	9	Integral Dist.	2^{704}	small	[16]
Grøstl-512	11	Trunc. Diff. Dist.	2^{640}	2^{64}	[26]
Grøstl-512	10	Integral Dist.	2^{513}	small	this paper

6 Conclusion

In this paper, we analyzed some SHA-3 candidates (Hamsi-256, LANE-256, Groestl-512) in regard of Integral properties. Due to a slow diffusion of the linear part of the proposed candidates, integral properties could be exhibited for a number of rounds greater than expected. We sum up our results and the related works concerning distinguishers on the compression functions of the SHA3 candidates in Table 1 where Prob. Int. Dist. means Probabilistic Integral Dinstinguisher.

References

1. Anderson, R.J., Biham, E., Knudsen, L.R.: Serpent : A proposal for the advanced encryption standard. In: The First Advanced Encryption Standard Candidate Conference. N.I.S.T. (1998), `http://www.cl.cam.ac.uk/~rja14/serpent.html`
2. Aumasson, J.-P., Käsper, E., Knudsen, L.R., Matusiewicz, K., Ødegård, R., Peyrin, T., Schläffer, M.: Distinguishers for the compression function and output transformation of hamsi-256. Cryptology ePrint Archive, Report 2010/091, to appear in ACISP 2010 (2010), `http://eprint.iacr.org/`
3. Biryukov, A., De Cannière, C., Dellkrantz, G.: Cryptanalysis of safer++. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 195–211. Springer, Heidelberg (2003)
4. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
5. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
6. Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 394–405. Springer, Heidelberg (2001)
7. Boura, C., Canteaut, A.: A zero-sum property for the keccak-f permutation with 18 rounds. NIST mailing list (2010)

8. Calik, C., Turan, M.S.: Message recovery and pseudo-preimage attacks on the compression function of hamsi-256. Cryptology ePrint Archive, Report 2010/057
9. Collard, B., Standaert, F.-X.: A statistical saturation attack against the block cipher present. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
10. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
11. Daemen, J., Rijmen, V.: Aes proposal: Rijndael. In: The First Advanced Encryption Standard Candidate Conference. N.I.S.T. (1998)
12. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer, Heidelberg (2002)
13. D'Halluin, C., Bijnens, G., Rijmen, V., Preneel, B.: Attack on six rounds of crypton. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 46–59. Springer, Heidelberg (1999)
14. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved cryptanalysis of rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)
15. Galice, S., Minier, M.: Improving integral attacks against rijndael-256 up to 9 rounds. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 1–15. Springer, Heidelberg (2008)
16. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl – a sha-3 candidate. Submission to NIST (2008)
17. Gilbert, H., Peyrin, T.: Super-sbox cryptanalysis: Improved attacks for aes-like permutations. Cryptology ePrint Archive, Report 2009/531, to appear at FSE 2010 (2009), http://eprint.iacr.org/
18. Indesteege, S.: The lane hash function. Submission to NIST (2008)
19. Nakahara Jr., J., de Freitas, D.S., Phan, R.C.-W.: New multiset attacks on rijndael with large blocks. In: Dawson, E., Vaudenay, S. (eds.) MYCRYPT 2005. LNCS, vol. 3715, pp. 277–295. Springer, Heidelberg (2005)
20. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
21. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
22. Küçük, Ö.: The hash function hamsi. Submission to NIST (updated) (2009)
23. Matusiewicz, K., Naya-Plasencia, M., Nikolic, I., Sasaki, Y., Schläffer, M.: Rebound attack on the full lane compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 106–125. Springer, Heidelberg (2009)
24. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Rebound attacks on the reduced grøstl hash function. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 350–365. Springer, Heidelberg (2010)
25. Minier, M., Phan, R.C.-W., Pousse, B.: Distinguishers for ciphers and known key attack against rijndael with large blocks. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 60–76. Springer, Heidelberg (2009)
26. Peyrin, T.: Improved differential attacks for echo and grøstl. Cryptology ePrint Archive, Report 2010/223, to appear in Crypto 2010 (2010), http://eprint.iacr.org/

A The Integral LANE Properties

Here are all the figures for a better illustration of Section 4.

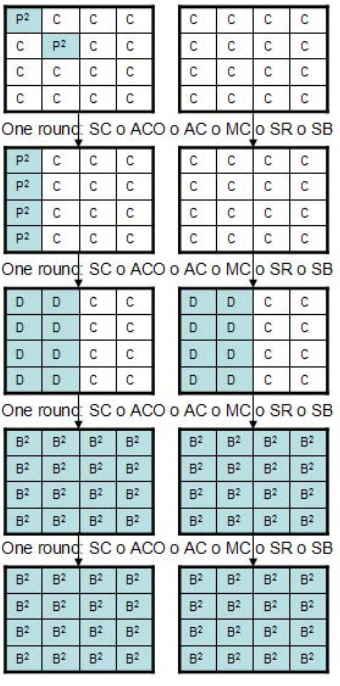

Fig. 9. The 2nd order Integral property on 4 forward rounds of P_i

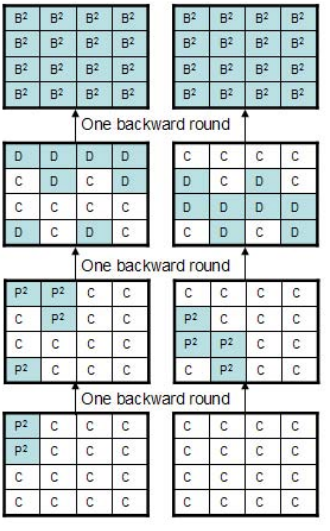

Fig. 10. The 2nd order Integral property on 4 backward rounds of P_i or Q_i

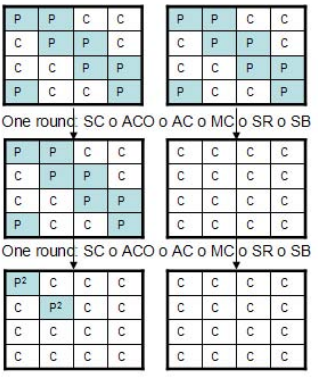

Fig. 11. The 8th order Integral extension on 4 forward rounds of P_i

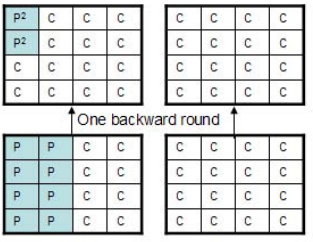

Fig. 12. The 8th order Integral extension on 3 backward rounds of P_i

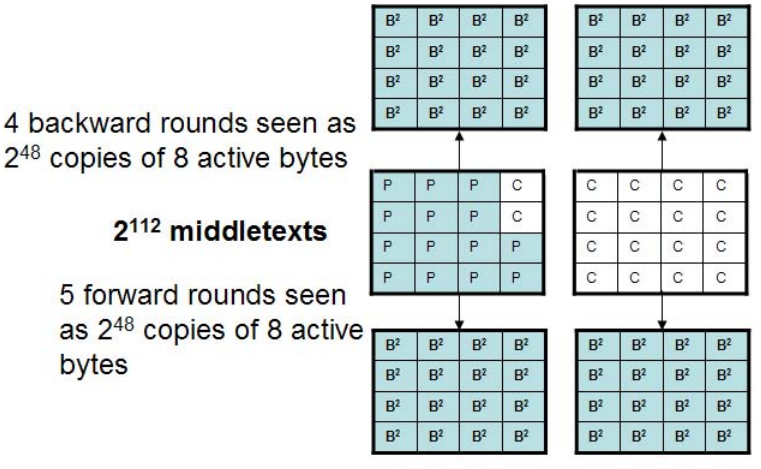

Fig. 13. The 9 rounds integral property using 2^{112} middletexts

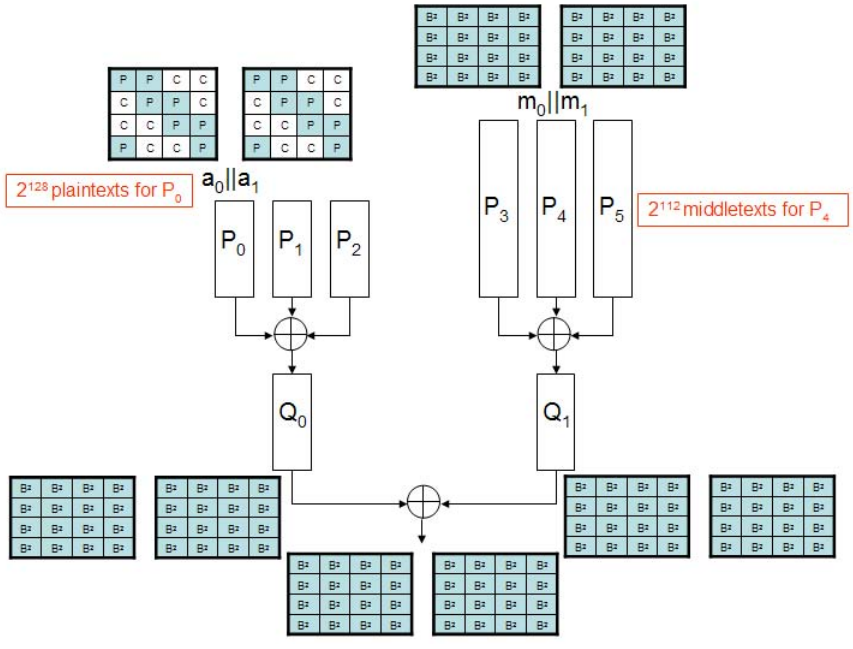

Fig. 14. The complete attack on the compression function where P_0, P_1 and P_2 are limited to 3 rounds using 2^{240} texts